Ingeniería Técnica en Informática de Gestión
Universidad Carlos III de Madrid

20/04/2015

Final Project

Answer2Pass Pro, multiplatform evaluation game

Author
Hugo Ismael Cornejo Ferradal

Directors
José María de Fuentes García-Romero de Tejada
Lorena González Manzano

Abstract

This Final Project covers the creation of an elearning software platform which enables teachers to evaluate their students through quizzes. Every quizz is different, showing different board sizes, category colours, randomly picked questions and answers and joker squares. In addition to of this, students can send their own questions to the platform.

Thanks to the integration with *Facebook* the students can log in into the platform using their credentials and during the course of the game can send questions to other students using *Facebook DM*.

On the teachers' side the platform offers a few tools to help with the real everyday use, as the ability to create a hierarchy of subjects and topics or a feature to export all the results to a spreadsheet.

Resumen

Este Proyecto Final de Carrera consiste en la creación de una plataforma de aprendizaje que permita a profesores evaluar a sus alumnos a través de juegos de preguntas y respuestas. Cada juego es diferente a los demás, con tableros de distintos tamaños, colores por categoría, preguntas y respuestas aleatorias y casillas de la suerte. Además los propios alumnos pueden enviar preguntas de creación propia a la plataforma.

Gracias a la integración con *Facebook* los alumnos pueden registrarse en la plataforma utilizando sus credenciales de la red social y durante el transcurso de los juegos pueden enviar preguntas a otros compañeros usando la herramienta de mensajes directos de *Facebook*.

En el lado de los profesores se han diseñado varias herramientas para facilitar su uso en un entorno real, como la posibilidad de tener jerarquías de asignaturas y temas o la exportación de resultados a hoja de cálculo.

Acknowledgements

My sincerest gratitude to Chema, he is the reason why this Project is a reality. I abandoned it (and him) many times, during many years, but Chema has been always willing to chase me just to offer me second chances. In all honesty, one of the finest teachers I have ever had the pleasure to deal with.

Many thanks to Lorena too, because of her this document is what it is and not a dysfunctional monster full of browsers' screenshots.

A final note to thank my good friend Pablo for all his help with technical stuff and my partner Lucia for dealing with a boyfriend glued to a computer screen during all those long hours and weekends.

Mamá, ¿recuerdas que durante todos estos años siempre que me lo has preguntado te he dicho que me daba igual no acabar el Proyecto ni tener el título?

Bueno, hoy puedo confesar que te estaba mintiendo.

Espero que esto lo compense y estés tan orgullosa de mí como yo estoy de ti.

Table of contents

1. Introduction

Everybody knows about the origins of the internet and, during the last ten years, the creation and spread of the *Social Network Services*. Running in parallel during the decade a massive adoption of new devices and the availability of mobile broadband had produce an environment particularly appealing for disruptive markets.

Examples of this are pretty much everywhere these days:

- **Wikipedia** [1]. An always updated, free access enciclopedia that has made useless traditional paper editions like the Encyclopædia Britannica, edited for the last time on paper in 2012.
- **Spotify** [2] **and iTunes** [3]. After the explosion of the *MP3* format some companies decided to use the same medium to sell music, instead of keep selling physical discs. Today physical albums are a product for collectors, specially vinyl [4].
- **Netflix** [5]. A video on demand service in opposition to the traditional TV channel broadcasting system, allowing customers to cherry pick what and how they want to see contents (TV, computer, tablet, phone).
- **EventBrite** [6]. A do-it-yourself platform to create events and sell digital tickets.
- **AirBnb** [7]. A system to rent flats between individuals during short periods in direct competition with hotels.
- **Uber** [8]. A handy app to operate and hire private cars for taxi services.

1.1. Motivation

It seems that the methods of evaluation are still stucked in the old traditional ways, basically papers and exams. In countries as Finland they are even considering abandon teaching subjects [9].

Surprisingly the technology have not disrupted the education yet and that is something strange because young people are precisely one of the easiest markets in terms of innovation, they have less prejudices towards new technologies and their opportunity costs are lower.

Said this, the main motivation of this Final Project is to create a piece of software that enables new ways of learning, evaluation and interaction. Not just between teachers and students but also amongst students.

1.2. The original Answer2Pass objectives

This Final Project is meant to extend the work of Vicente Domínguez Martín's **Answer2Pass**. UC3M Final Project developed in 2012 [10].

These were the original objectives:

- Create a web application to allow teachers to evaluate students using quizzes.
- The web application must allow students to play an undefined number of games answering questions over a colour changing board.
- The student could use jokers, as sending a question to a classmate or creating their own question.
- All the scores must be kept and shown in rankings.
- Teacher must manage students accounts and be able to create questions and moderate the questions that students send to the platform.

1.3. Objectives

The main goal of this Final Project is to develop a web application, **Answer2Pass Pro**, built on top of the achievements of the original **Answer2Pass** while adding some interesting features.

- The application should allow multiple and customisable subjects. Each subject would be handled by a coordinator while an administrator is responsible of managing those coordinators through a user role system.
- New board sizes (small, medium and large) and the ability to set more than three topics in a board.
- Connect the number of squares that the student moves forward after every right answer with the difficulty of the question. Harder questions deserve bigger rewards.
- Each and every board can set their own access rules, allowing students to play during a period of time trying to improve their scores or restricting the game to just one opportunity per student.
- Once the board is closed (after a term, for example) the access is blocked for students while coordinators and administrators can check all the scores, export the results to a spreadsheet and clean the student lists for the next term.

1.4. Structure of this document

This document is structured in different chapters with the following contents

1. **Introduction (this one)**. A proper introduction to the context, motivation and objectives for the Final Project. Intended to be readable by a non-technical audience without major doubts.
2. **State of the art**. A full chapter to identify and analyse competitors, to know exactly what is on the market.
3. **Analysis**. A technical approach to the nature of the problem to solve and the available solutions.
4. **Design**. A highly technical chapter explaining the solution developed.
5. **Implementation**. A chapter about the implementation and small but relevante details of the actual coding and creation of the application.
6. **Project planning**. Because a software project without a plan or financial analysis is the first step to failure, this chapter tries to avoid that.
7. **Conclusions**. A summary of all the lessons learnt during the realisation of this Final Project.
8. **User manual**. The back office of the app requires some clear indications to be used, this manual covers that part.
9. **Reference**. All the information sources consulted.

2. State of the art

2.1. Quiz

Before even talking about technology is interesting to think about the concept of quiz, a game of the mind in which different players try to answer questions correctly in order to get points or advance in a board.

The word itself it has been around for almost 250 years [11] and that is not surprising because playing quiz both by yourself or against others is a rather fun way not just to show off knowledge in front of your friends and family but also to learn new content and fix incorrect concepts.

Due to its proven ability to entertain quiz were adapted to radio and TV [12] almost 100 years ago and 50 [13] in the case of computer games. The fluid shape of quiz makes it a kind of game really flexible to be adapted to different technologies and scenarios, from pub quiz nights to, probably, smart watches apps.

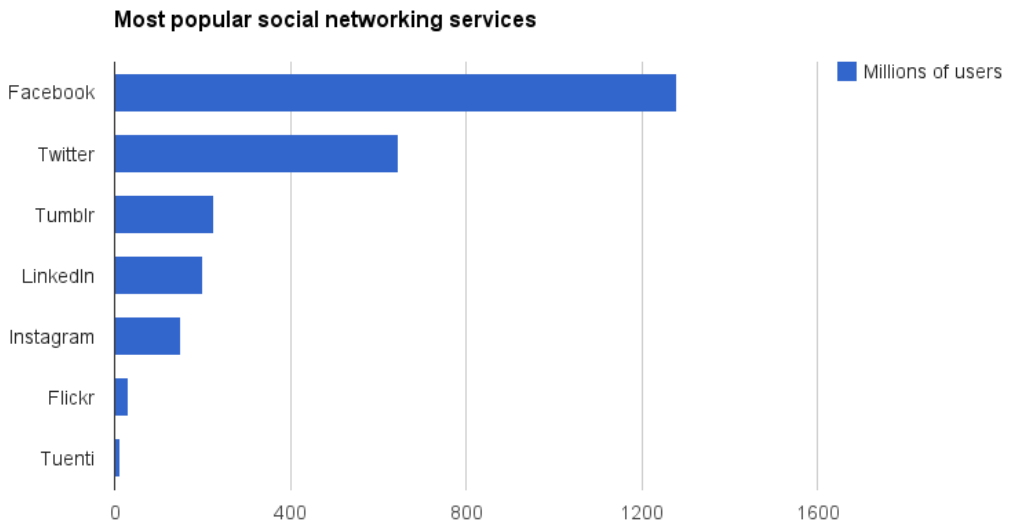
2.2. Social networking services

Despite the fact that the first *Social Networking Service* (SNS) was born on 1997 [14] the true popularity of *SNS* has emerged during the last decade. Particularly after the introduction of smartphones and affordable wireless internet access, *SNS* has quickly become the bread and butter of the Internet.

From *Facebook* to *Twitter* or *LinkedIn*, *SNS* are by far the killer feature of many mobile devices and the app of choice for millions of people to communicate with their friends, instead of using *Email* or messaging apps.

2.2.1. Usage around the world

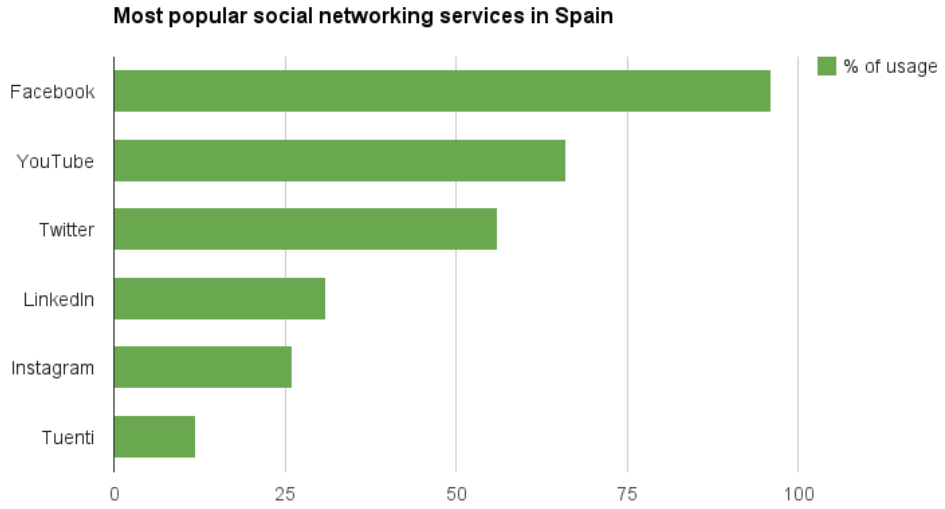
If we look at the *Global Alexa page ranking* as indicator of popularity of a network, and we put together the number of users of each of those networks [15] we have an overview in which is extremely clear that *Facebook* is the strongest player with roughly twice as many users as their next competitor, *Twitter*.



2.2.2. Usage in Spain

Because of their very nature, social networking services are particularly difficult to export between countries and that is why many countries have their own *SNS*, typically competing with the long tail international players.

Spain is not an exception to that. According to the *IAB Spain* (Interactive Advertising Bureau) [16], *Tuenti* is still used by 1 out of 9 internet users in the country. However, their use has decreased by an astonishing 50% just in 2014 while the rest of competitors are growing at a 2~3% each year.



2.2.3. Facebook

Founded in 2004 *Facebook* is the biggest *SNS* on the market [17], present in pretty much every country in the world and translated to more than 70 languages.

The core of *Facebook* are user profiles (basically a stream of content as texts, links, photos and videos) and direct messages between users. On top of that there are lots of apps (thanks to the *API* [18]) and extra features that take advantage of the user connections to help them find their friends in other platforms, log in easily, and so on.

2.2.4. Twitter

Created in 2006 *Twitter* is the most important *SNS* on mobile devices. From the very beginning they focused only in sharing short messages (140 characters tops, called 'tweets'), which works really well on phones, where the quality and speed of access to the internet is not always optimal.

As well as *Facebook*, *Twitter* provides a full *API* [19] to let developers to build apps on top of the login and networking layers.

2.2.5. LinkedIn

In contrast with *Facebook* and *Twitter*, *LinkedIn* is not a general *SNS*, is business orientated. Launched in 2003 (a year before *Facebook*) is the biggest place to keep a professional profile (a digital equivalent of a business card) and to keep track of your professional contacts.

Usually students don't have *LinkedIn* profiles until they finish their studies and join the workforce but companies as the Spanish *Tyba* [20] are trying to close that gap allowing students to create profiles not just based on professional experience and studies (that they don't have yet) but on interests and references. Something to keep an eye on the future.

2.3. Similar software

There are some software products already on the market offering similar features to the objectives of this Final Project. There are literally hundreds of products claiming to be quizzes solutions, for obvious reasons this document only analyses the most relevant.

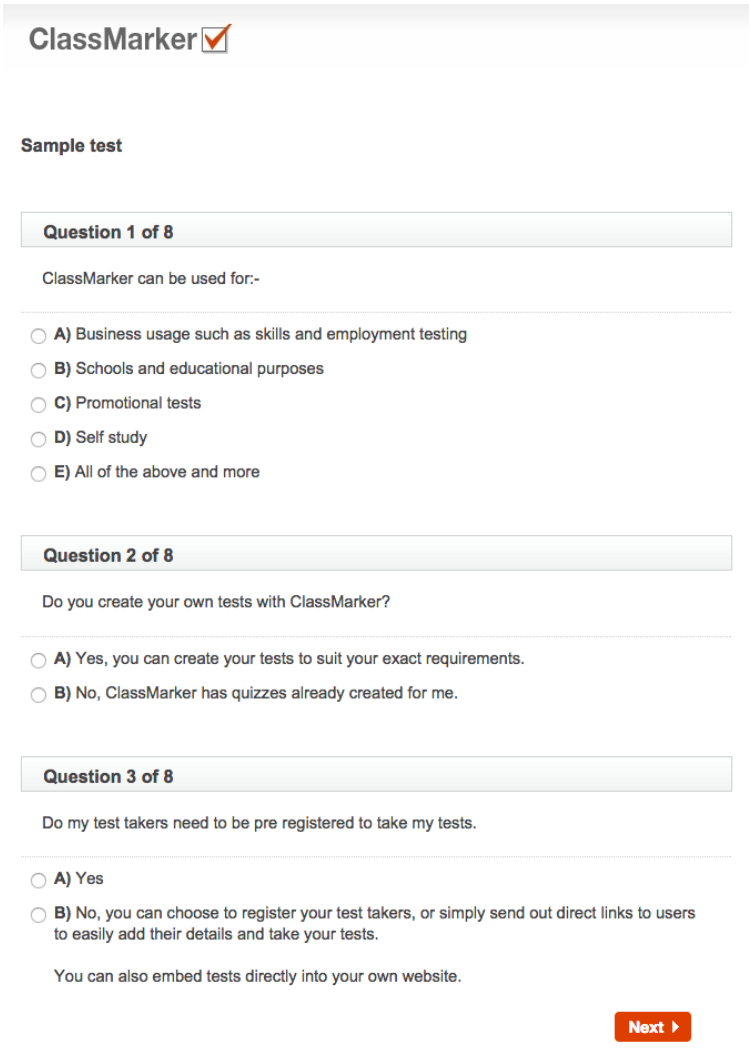
This section provides an overview of the origins, features, requirements, strengths and weaknesses of these competitors in order to reveal **Answer2Pass Pro** value and adjust its fit to market.

2.3.1. ClassMarker

ClassMarker [21] is a professional online testing tool for both business and education. The platform is entirely web-based and the test follows a *Responsive Design* front-end fleur, so they work perfectly in mobile devices too.

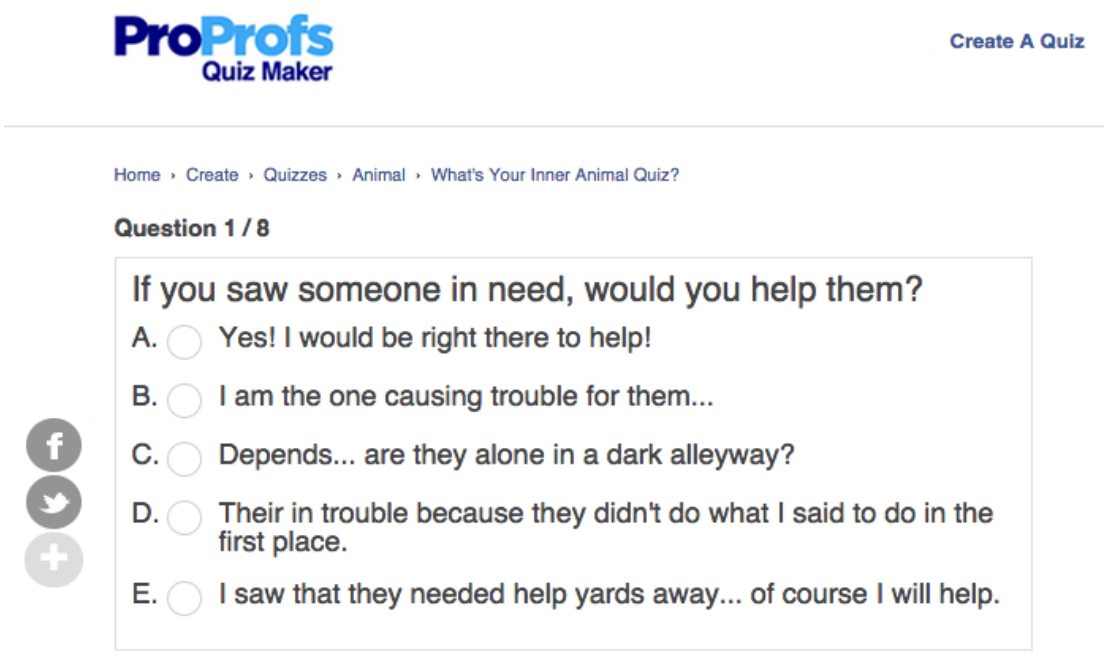
One interesting feature is the presence of an *API*, so developers can connect other apps and extend functionalities on top of the core, something particularly interesting to, for example, provide single sign on with the elearning platform of the university, etc.

The pricing follows a pay-as-you-go scheme starting at free up to 100 quizzes a month to 35 €/month for 1000 quizzes a month.


ClassMarker
Sample test
Question 1 of 8
ClassMarker can be used for:-
A) Business usage such as skills and employment testing
B) Schools and educational purposes
C) Promotional tests
D) Self study
E) All of the above and more
Question 2 of 8
Do you create your own tests with ClassMarker?
A) Yes, you can create your tests to suit your exact requirements.
B) No, ClassMarker has quizzes already created for me.
Question 3 of 8
Do my test takers need to be pre registered to take my tests.
A) Yes
B) No, you can choose to register your test takers, or simply send out direct links to users to easily add their details and take your tests.
You can also embed tests directly into your own website.
Next

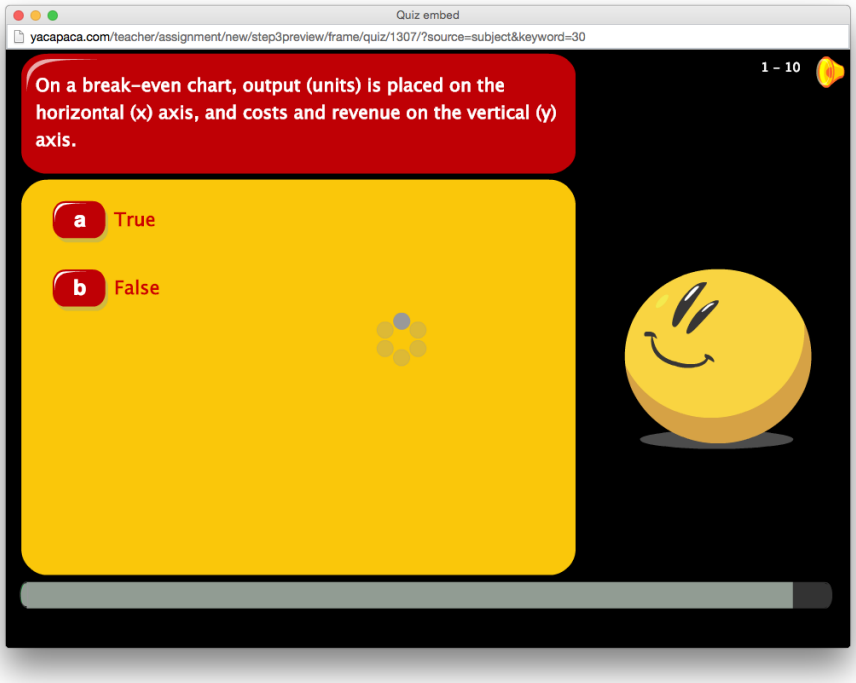
2.3.2. ProProfs

ProProfs [22] offers a professional web-based solution for quizzes with a price range from 35 €/month to 180 €/month, depending on the number of users and questions.



2.3.3. Yacapaca!

Yacapaca! [23] presents an *Adobe Flash*-based solution for quizzes and evaluation in schools. The structure follows the path Country > Subject > Syllabus > Topic > Quiz.

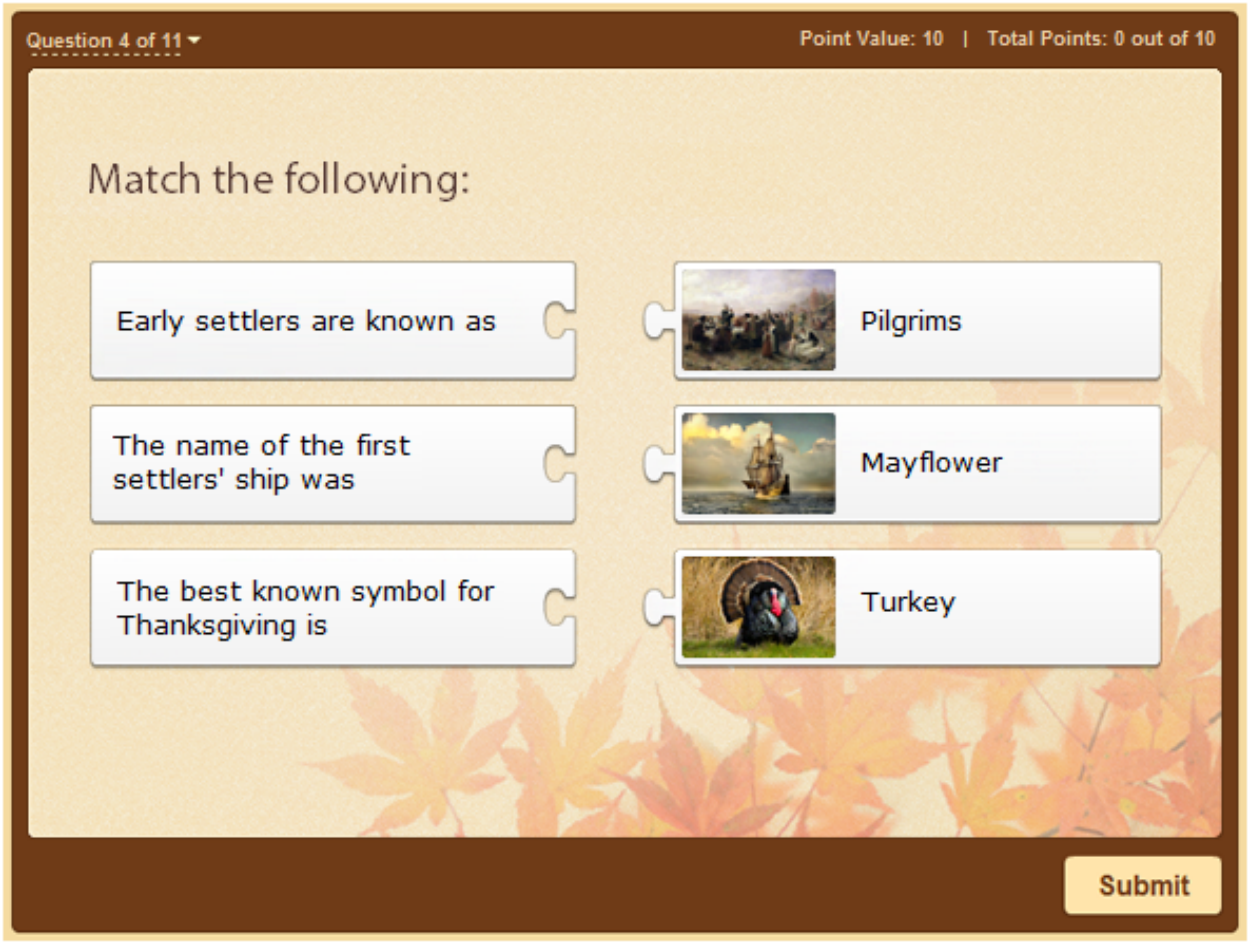


One of the most interesting features of this software is that allow students to review quizzes, that way teachers can improve the contents.

It is *SAAS* prices range from 140 €/year to 1,400 €/year depending on the number of teachers, quizzes per month and support availability.

2.3.4. iSpring QuizMaker

QuizMaker [24] seems a very robust quiz solution with lots of attention paid to help teachers to track the results and importing questions straight from a spreadsheet.



The quizzes are based on *Flash* and *HTML5* templates (with many different quizz styles, flexible layout options and customisable colour themes) while the back office is a standalone *Windows* app.

iSpring offers licensing for universities starting at 200 €. That does not include the cost of hosting the final tests.

2.3.5. Kahoot!

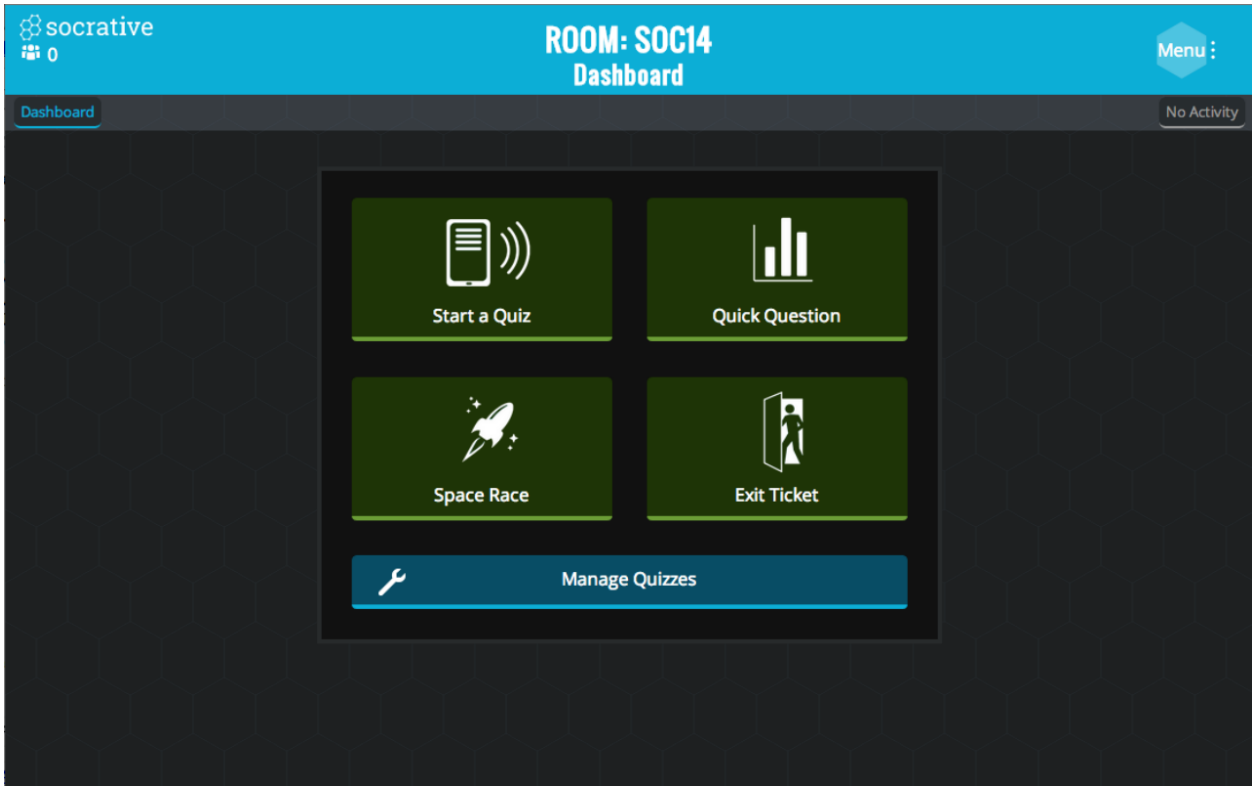
Kahoot! [25] is a game-based classroom response system, in other words, a multiplayer game designed to be played by students using their own phones while the board and questions appear in the class projector interactive whiteboard.

This product is really well designed, easy to use thanks to the utilisation of *Responsive Design* (a front-end development technique which allows browsers of all sizes and capabilities to operate websites with the very same *HTML* code) and a clever approach of making the students engage each others while playing in person, something usually forgotten in digital tools.

2.3.6. Socrative

Socrative [26] is a platform that allow teachers to present quizzes to students during the course of a lesson, similarly to *Kahoot!*.

It is free to use and supports different game styles, which helps to keep alive the students' interest.



2.3.7. Comparison of results

The following table compares all the analysed apps to discover if there is or not a market opportunity for **Answer2Pass Pro**.

	Class Marker	Pro Profs	Yacapaca!	Quiz Maker	Kahoot!	Socrative
Free to use						
Fun to play						
Students send questions						
Set dates						
Subject hierarchy						
Teacher hierarchy						
Social network integration						
Multiplatform						
Responsive Design						
Spreadsheet						

The data proves that there is not a single software product offering all the intended features for **Answer2Pass Pro**.

A pattern appears clearly, the apps that give power to the teachers to create complex hierarchies and question data banks (like *ClassMarker*) are not fun to play or not even games, while the ones that are fun to play and attractively

designed (like *Socrative* or *Kahoot!*) lack the features that are mandatory for a big school or university.

The conclusion is that there is an opportunity to make **Answer2Pass Pro** a product with a clear advantage over the rest of its competitors, covering on one hand all the features for a hierarchy of tutors, export to spreadsheet, availability dates and in the other hand a quiz fun to play, with a proper board and rewards; in other words, a game.

3. Analysis

3.1. Summary

The aim of this section is to provide a thorough analysis of the piece of software intended to develop as a result of this Final Project. The output of the analysis would be the required foundation to design the software properly.

The structure is the following:

- **Analysis of technologies and alternatives**: All the relevant aspects in terms of platform, languages, frameworks, *SNS*, storage and deploy solutions.
- **Technologies of choice**: A reasoned narration about the technologies chosen to build this Final Project.
- **Use case diagrams**: *UML* diagrams to show the different use cases that the application must cover.
- **Software requirements**: A comprehensive list of all the identified software requirements.
- **Testing plan**: A set of tests created to check that all the software requirements are properly implemented.

3.2. Analysis of technologies and alternatives

3.2.1. Platform

One of the requirements of this Final Project is to provide a multiplatform experience for all students. An app available for desktop computers (*Mac*, *Windows*, *Linux*, etc.) and for mobile devices too (*iPhone*, *Android*, *Windows Phone*, etc.).

This requirement instantly narrows the platform choice to build **Answer2Pass Pro** as a web application. Using well crafted *HTML* and *Responsive Design* [27] is perfectly reasonable to implement a one-size-fits-all solution, instead of building native apps for every platform (which would be a nightmare in developing and maintenance costs).

3.2.2. Web application frameworks

When building web applications is very common to start using a *Web Application Framework* (WAF) [28]. Frameworks usually take care of lots of the scaffolding tasks of an app (connectors to databases, templating, sessions management, etc.) enabling at the same time simple ways to integrate third-party libraries (authentication, social networks *API*, etc.).

Pretty much every major language provides web application frameworks, let's see an overview of the most popular at the moment.

3.2.2.1. C# & VB.NET

On the *Microsoft Windows* platform the king is *ASP.NET MVC Framework*. Its biggest strength is *Visual Studio*, one of the finest *Integrated Development Environments* (IDE) available to developers on *Windows*.

3.2.2.2. Perl

In *Perl* the best *WAF* is *Catalyst*, open-source and really close to the *Model-view-controller* (MVC) architecture. Probably its biggest strength lays on the distribution channel, *CPAN*, where any user can download and install *Catalyst* with a single command line.

3.2.2.3. Java

In the huge world of *Java* the most relevant *WAF* are *Spring* (heavily inspired in *Catalyst*), *Play* and *Google Web Toolkit* (GWT). All of them support the *MVC* paradigm, asynchronous petitions, internationalisation, localisation, security and template frameworks with, of course, form validation.

3.2.2.4. PHP

PHP is by far the most prolific language in terms of web application frameworks catalog, with great examples of *MVC* solutions as *Zend Framework*, *CakePHP*, *Symfony* or *CodeIgniter*.

The biggest strength of the platform lays in the huge community of developers and the variety of affordable choices to host *PHP* projects on the internet.

3.2.2.5. Python

In this language the choice de facto is *Django*, followed by *pylons*. *Django* is heavily oriented towards database-driven applications, paying special attention to rapid developments through pluggability of components and reusability.

3.2.2.6. Ruby

The popularity of *Ruby* exploded with the birth of *Ruby on Rails*. Inspired in *Catalyst* follows the *MVC* architecture and the 'convention over configuration' paradigm, just the opposite to *Django's* 'explicit is better than implicit'.

3.2.2.7. JavaScript

In client side there are some *WAF* too. Particularly during the last five years many players have been born: e.g. *AngularJS*, *Ember.js* and *Node.js*. The approach in these frameworks is usually to employ the *DOM* user interfaces as a mean to connect other software components, while keeping the business logic independent.

3.2.3. SNS platforms

This section aims to analyse the major *Social Network Services* available for the sharing and collaborative features required for **Answer2Pass Pro**.

3.2.3.1. Facebook

Facebook offers a great deal of *APIs* for developers to integrate apps with pretty much every *Facebook* feature, helping to this mission with a very complete developers site, packed with documentation and videos.

3.2.3.2. Twitter

Twitter gives access to absolute every feature through their *API*, which is interesting because *Twitter* official clients (both web and apps) are really just another client consuming the same *API* than the rest of third-party apps.

3.2.3.3. LinkedIn

LinkedIn offers too a simple *API* [29] to let developers use *LinkedIn* Login using a *REST* interface. The focus, of course, is on the professional-oriented apps.

3.2.4. Storage technologies

Answer2Pass Pro requires some kind of persistence layer to keep stored the boards, questions, answers and, of course, students. This section reviews the most popular choices for it.

3.2.4.1. SQLite

SQLite is probably the easiest way to have a fully working *SQL* database. The whole storage relies in a simple plain file, which makes it an impossible solution for apps that need concurrent writing. It is habitually used as internal storage on stand-alone apps, where just one process takes care of all the writing operations.

3.2.4.2. PostgreSQL

PostgreSQL implements almost all the *SQL:2011* standard, fully *ACID* and transactional compliant. It is open source, fact that has allowed the forking of the database in many other derived products.

3.2.4.3. MySQL

MySQL, owned by *Oracle*, is an open source database, core of the *LAMP* stack and used in many products, from software packages as *WordPress* and *phpBB* or *Drupal* to huge sites as *Facebook* and *Flickr*.

3.2.4.4. MongoDB

MongoDB is a *NoSQL* database, based on documents in *JSON* with dynamic schemas. Its main advantage over other databases is the horizontal scalability based on sharding.

3.2.5. Deploy

The best way to deploy a product as **Answer2Pass Pro** is to rely on platforms as a service cloud computing solutions. Host this kind of app in an independent or shared server can create serious trouble in terms of scalability once the maximum bandwidth of the provider is fully consumed.

In cloud computing, huge networks of servers, take customers applications and run them on sandboxed environments. The approach is more restrictive but, also, much more reliable.

Another advantage is that instead of paying a flat fee for hosting the app all this products follow a pay-as-you-go scheme, the more resources a customer app consumes the bigger the invoice.

3.2.5.1. Google Cloud Platform

Google offers its enormous computing power to let developers send up to 7 billion requests per day [30].

Many languages are accepted in their instances, *Python*, *PHP* and *Java*, while the access to storage and both *SQL* and *NoSQL* databases.

The bigger toll is that applications must be designed for the *Google Cloud Platform* infrastructure, because of this it is not easy to change providers once the application is on production.

3.2.5.2. Amazon Web Services

The suite of *Amazon* services for the web (AWS) [31], particularly *Amazon Elastic Compute Cloud* (EC2), is basically a huge cluster of virtual machines (replicated in

eleven geographical regions) where developers can run instances of *Linux*, *Windows* or *FreeBSD* operating systems.

The best feature of *AWS* is the independance of the virtual machines over the platform provided, which avoids any future lock in.

3.2.5.3. Heroku

Heroku [32] abstracts developers of the hardware using created concepts as *dynos* and a seamless integration with *GIT* and *PostgreSQL*. It is not the most affordable solution out there but it is a good place to start developing a web application and, once is ready for production, just pay a little bit extra to scale more dynos or database rows.

3.2.5.4. Rackspace

Rackspace's main approach [33] is identical to *Amazon Web Services*, virtual machines where developers can run whatever they want. The main difference is that, apart of the virtual machine solution they offer dedicated servers as well, an interesting feature when the nature of the web application data needs extra security in order to be compliant with high risk legality, etc.

3.2.5.5. SoftLayer

SoftLayer [34] business is exactly the same as *Rackspace*, both real and virtual servers are available.

3.2.5.6. Nitrous

Nitrous [35] is an interesting mix of Web *IDE* with a cloud infrastructure. Thanks to an abstraction scheme based on 'boxes', it is really easy to start a server with a fully functional *IP* gateway or domain and develop the web application right on the same server in which it could eventually be in production.

Once the code is ready *Nitrous* provides tutorials of how to deploy the apps to more serious platforms, like *Heroku* or *Google Cloud Platform*.

3.3. Technologies of choice

The chosen language and platform would be ***Python* and *Django***. The main reason is that the developer of this Final Project has very little experience on that language and he wants to learn about it because of the promising future of it.

In terms of social network service the chosen provider is ***Facebook*** because it is the most ubiquitous provider in the world and the *API* fits perfectly for the Final Project purposes.

The storage solution was ***SQLite*** during development time, switching to ***MySQL*** for the deploy and production life of the web application. For the nature of this Final Project would be the same to use *PostgreSQL* or *MongoDB*, because there is no use of *ACID* or triggers, just plain storage. *Django* abstracts the access to data in such an elegant way that changing the engine is as simple as adding two lines of configuration code.

The deploy will be done on ***Nitrous*** running *Apache*, mainly because of its optimised *Python* boxes and user-friendly interfaces that allows to install new databases or servers in a matter of minutes instead of hours.

3.4. Use case diagrams

The most important matter of this section is the *UML* use case diagram. That way it is easy to understand which are the users' interaction with the system, as well as illustrate the relationship between the user and the different use cases.

In this Final Project there are three kind of users: student, tutor and administrator. There is an external system too, *Facebook*, that is used for log in to the platform as well as granting access to social features like sending a question to a friend.

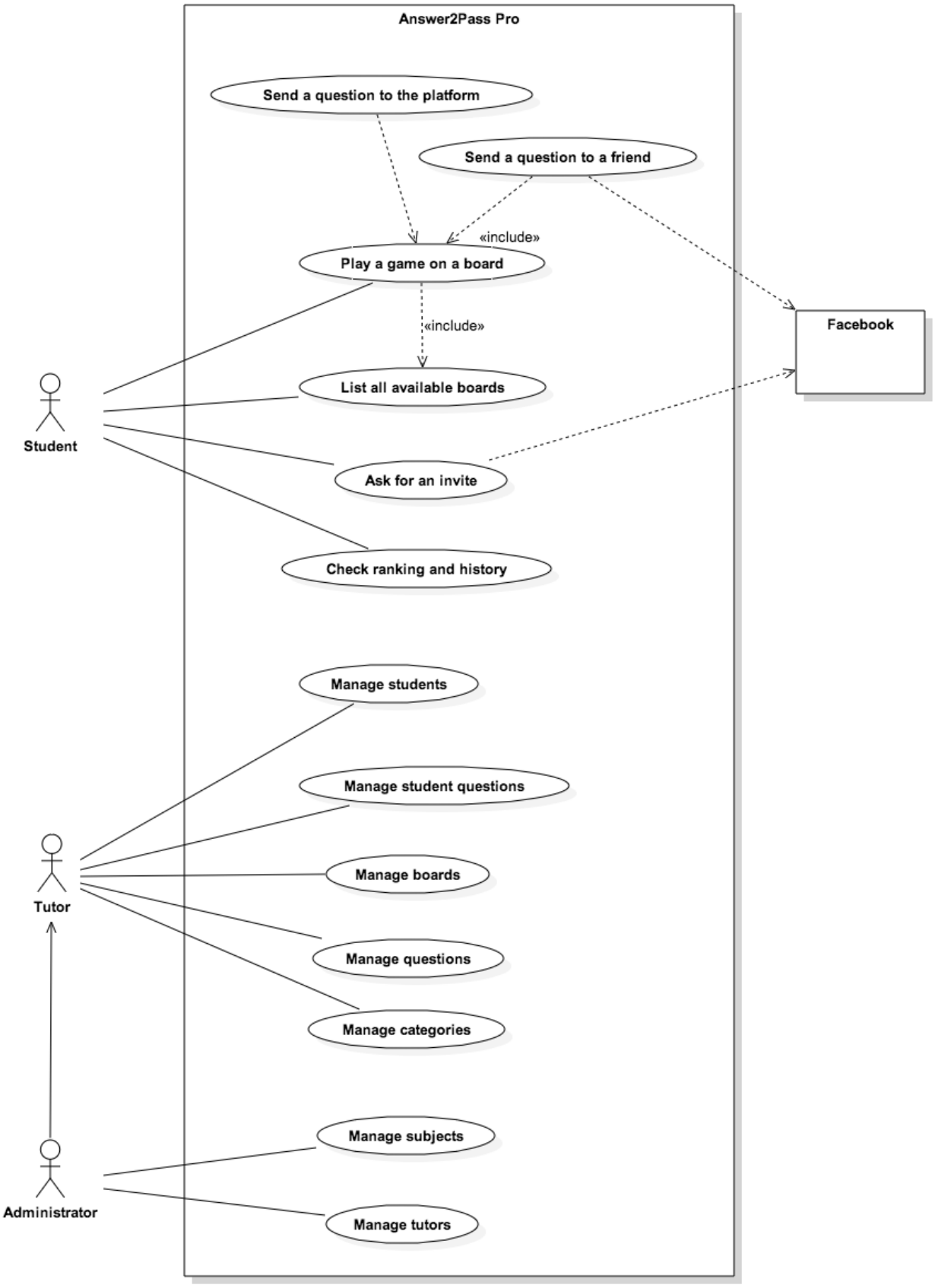
Three kind of actors:

- **Students**: UC3M's students, the players.
- **Tutors**: UC3M's teachers, they define the boards and majority of questions.
- **Administrator**: Takes care of managing the tutors accounts.

Overall, the use cases are:

- **Ask for an invite**: The game is meant to be private, just for authorised users. Because of this, students need to apply for an invite giving their full names and NIA.
- **List all available boards**: Before playing a game the student needs to see the full list of available boards.
- **Play a game on a board**: This is the core of the app, move through a board answering questions and scoring points to the finish line.
- **Check ranking and history**: A place to review old games and check who are the best players on the platform.
- **Send a question to a friend**: Pick a *Facebook* friend and send them a link to the app.
- **Send a question to the platform**: Students can send their own questions (and answers) to be reviewed by the tutors.
- **Manage students**: Ability to create, modify and delete student invites.
- **Manage students questions**: After students send their questions a tutor needs to check that everything is correct in order to add that question to the pool.

- **Manage boards**: Create and delete new boards, adjusting the categories, board size, scoring style and date availability.
- **Manage questions**: Create, modify and delete questions and answers, both single (true/false) and multiple choice.
- **Manage subjects**: Questions are related with subjects and subjects need to be created, modified and deleted.
- **Manage categories**: Exactly the same as in subjects but one hierarchy level higher.
- **Manage tutors**: Define tutor accounts and adjust which subject are they responsible for.


Answer2Pass Pro
Send a question to the platform
Send a question to a friend
«include»
Play a game on a board
«include»
List all available boards
Ask for an invite
Check ranking and history
Facebook
Student
Manage students
Manage student questions
Manage boards
Manage questions
Manage categories
Tutor
Manage subjects
Manage tutors
Administrator

3.5. Use case descriptions

In this section all the different use cases are detailed, showing preconditions, postconditions and flows for each of them.

The use cases related with the members of staff are based on *Django Suit* [36], with a different admin plugin the flow would be different.

ID	UC-01
Name	**Ask for an invite**
Description	Register the student as pending for an invite
Actors	Student
Preconditions	Student has a *Facebook* account and credentials
Postconditions	The invite request is saved
Flow	1. Student clicks on the *Facebook* login button 2. Student authorizes *Answer2Pass Pro* on *Facebook* 3. Student fills the form with Name and NIA 4. Student clicks the submit button

ID	UC-02
Name	**List all available boards**
Description	Show buttons for each available board
Actors	Student
Preconditions	Student has logged in into the platform
Postconditions	The list of boards is showed
Flow	1. Student clicks on the Boards button

ID	UC-03
Name	**Play a game on a board**
Description	A full quizz, answering questions and moving over the board until the finish line
Actors	Student
Preconditions	Student has logged in into the platform
Postconditions	The game is over and the score is saved
Flow	1. Student clicks on one board from the list 2. Student clicks on the first available question 3. Student answers the question 4. Repeat 2 and 3 until getting to the finish line 5. Dismiss the congratulations screen

ID	UC-04
Name	**Check ranking and history**
Description	A place to review old games and check who are the best players on the platform
Actors	Student
Preconditions	Student has logged in into the platform Some games have been finished before
Postconditions	Two rankings and the student history are shown
Flow	1. Student clicks on the Ranking button

ID	UC-05
Name	**Send a question to a friend**
Description	Players can send questions to their friends through *Facebook Direct Messages* (DM) feature
Actors	Student
Preconditions	Student has logged in into the platform Student is playing a board and next question is Extra The Extra random category is of this type
Postconditions	The selected friend receives a *DM* with a link to the platform
Flow	1. Student clicks on the Extra square 2. *Facebook* launches with the *DM* dialog 3. Student picks a friend and clicks Send 4. Go back to the board with a new square ready to play

ID	UC-06
Name	**Send a question to the platform**
Description	Players can send their own questions to the platform
Actors	Student
Preconditions	Student has logged in into the platform Student is playing a board and next question is Extra The Extra random category is of this type
Postconditions	The platform receives the new question and answers
Flow	1. Student clicks on the Extra square 2. A form is shown 3. Student fills all the fields with the question and answers 4. Student clicks Send 5. Go back to the board with a new square ready to play

ID	UC-07
Name	**Manage students**
Description	Members of staff can create, modify or delete student invites
Actors	Tutor
Preconditions	Tutor has logged in into the platform
Postconditions	The changes are committed to the platform
Flow	1. Tutor clicks on the Students menu 2. Tutor makes changes through the admin *UI* 3. Tutor saves changes

ID	UC-08
Name	**Manage students questions**
Description	Members of staff can create, modify or delete student invites
Actors	Tutor
Preconditions	Tutor has logged in into the platform There are pending questions from students
Postconditions	The changes are committed to the platform
Flow	1. Tutor clicks on the Questions menu 2. Tutor makes changes through the admin *UI* 3. Tutor saves changes

ID	UC-09
Name	**Manage boards**
Description	Members of staff can create or delete boards
Actors	Tutor
Preconditions	Tutor has logged in into the platform
Postconditions	The changes are committed to the platform
Flow	1. Tutor clicks on the Panels menu 2. Tutor makes changes through the admin *UI* 3. Tutor saves changes

ID	UC-10
Name	**Manage questions**
Description	Members of staff can create, modify or delete questions and answers
Actors	Tutor
Preconditions	Tutor has logged in into the platform
Postconditions	The changes are committed to the platform
Flow	1. Tutor clicks on the Questions menu 2. Tutor makes changes through the admin *UI* 3. Tutor saves changes

ID	UC-11
Name	**Manage categories**
Description	Members of staff can create, modify or delete categories
Actors	Tutor
Preconditions	Tutor has logged in into the platform
Postconditions	The changes are committed to the platform
Flow	1. Tutor clicks on the Categories menu 2. Tutor makes changes through the admin *UI* 3. Tutor saves changes

ID	UC-12
Name	**Manage subjects**
Description	Administrators can create, modify or delete subjects
Actors	Administrator
Preconditions	Administrator has logged in into the platform
Postconditions	The changes are committed to the platform
Flow	1. Administrator clicks on the Subjects menu 2. Administrator makes changes through the admin *UI* 3. Administrator saves changes

ID	UC-13
Name	**Manage tutors**
Description	Administrators can create, modify or delete tutor accounts
Actors	Administrator
Preconditions	Administrator has logged in into the platform
Postconditions	The changes are committed to the platform
Flow	1. Administrator clicks on the Users menu 2. Administrator makes changes through the admin *UI* 3. Administrator saves changes

3.6. Software requirements

This section of the document covers all the software requirements for the web application, and defines what are the expected behaviours.

3.6.1. Format

The format used in the subsequent tables is the following:

- **ID**: A unique key to be able to discern without doubt one requirement from the other.
- **Name**: Overview of the requirement.
- **Description**: Requirement in detail.
- **Priority**: How important is to fulfill the requirement; low, medium or high.
- **Verifiability**: Reference to the test or tests that proves the correct implementation.

3.6.2. Functional requirements

This requirements will help to understand the web application and its behaviour.

3.6.2.1. Operational functional requirements

ID	Name	Description	Priority	Verifiability
FR-01	Ask for an invite	The app needs to allow the user to send an invite request to the staff team.	High	AT-05
FR-02	Accept an invite	Members of staff should be able to accept a student invite	High	AT-13

		request.		
FR-03	Decline an invite	Members of staff should be able to decline a student invite request.	High	AT-13
FR-04	Log in as student	Once the student has been approved as a legitimate user the app needs to let them log in.	High	AT-01
FR-05	Log in as admin	The administrator should be able to log in in the back office.	High	AT-13
FR-06	Log in as tutor	The tutors should be able to log in in the back office.	High	AT-13
FR-07	Deny access	No one should access the app until their invite ticket is approved.	High	AT-05
FR-08	List available boards	The app must allow students to see all the boards available at any time.	High	AT-06
FR-09	Open a board	Students must be able to open a board.	High	AT-07
FR-10	Go to a square	Students must be able to go to a square and reveal the associated random question.	High	AT-08
FR-11	True/false questions	Students must answer if a statement is true or false.	High	AT-08
FR-12	Multiple choice questions	Students must choose one of the right answers to a given question.	High	AT-08
FR-13	Image questions	Questions can contain a contextual image.	Low	AT-08

FR-14	Answer correction	The app must inform the student if their answer was right or wrong.	High	AT-09 AT-10
FR-15	Question timer	The app must keep a timer of how long took the student to answer the question.	High	AT-11
FR-16	Question difficulty	The app must control the difficulty of each question.	High	AT-09
FR-17	Send a question to the platform	Students must be able to send a custom question to the platform when they run in an Extra square.	Low	AT-08
FR-18	Send a question to a friend	Students must be able to send a link to a *Facebook* friend when they run in an Extra square.	Low	AT-08
FR-19	Share score	The app must share the score of a game in the student *Facebook* timeline.	Low	AT-11
FR-20	Check ranking and history	Students must be able to check who are the best players on the platform and their own results.	Mid	AT-12
FR-21	List all student questions	Members of staff must be able to see the whole list of pending questions from students.	High	AT-13
FR-22	Approve student question	Members of staff must be able to correct and accept questions from the students.	Mid	AT-13
FR-23	Discard student question	Members of staff must be able to discard questions from the students.	Mid	AT-13

FR-24	List all boards	Members of staff must be able to see the list of all created boards.	High	AT-13
FR-25	Create a board	Members of staff must be able to create new boards.	High	AT-13
FR-26	Delete a board	Members of staff must be able to delete boards.	High	AT-13
FR-27	List all questions	Members of staff must be able to see all available questions of a category.	High	AT-13
FR-28	Create question	Members of staff must be able to create new questions.	High	AT-13
FR-29	Modify question	Members of staff must be able to modify questions.	High	AT-13
FR-30	Delete question	Members of staff must be able to delete questions.	High	AT-13
FR-31	List all categories	Members of staff must be able to see all the categories of a subject.	High	AT-13
FR-32	Create category	Members of staff must be able to create new categories.	High	AT-13
FR-33	Modify category	Members of staff must be able to modify categories.	High	AT-13
FR-34	Delete category	Members of staff must be able to delete categories.	High	AT-13
FR-35	List all subjects	Members of staff must be able to see the list of all subjects.	High	AT-13
FR-36	Create subject	Members of staff must be able to create new subjects.	High	AT-13
FR-37	Modify subject	Members of staff must be	High	AT-13

		able to modify subjects.		
FR-38	Delete subject	Members of staff must be able to delete subjects.	High	AT-13
FR-39	List all tutors	The administrator must be able to see a list of all the tutors.	High	AT-13
FR-40	Create tutor account	The administrator must be able to create a new tutor account.	High	AT-13
FR-41	Modify tutor account	The administrator must be able to modify a tutor account.	High	AT-13
FR-42	Delete tutor account	The administrator must be able to delete a tutor account.	High	AT-13
FR-43	Export results	Members of staff must be able to export students results as a spreadsheet.	Mid	AT-13

3.6.2.2. Data functional requirements

ID	Name	Description	Priority	Verifiability
FR-44	Welcome page	Students must be welcomed by an easy login screen.	High	AT-01
FR-45	Invite page	The app must provide an easy form to get students' full names and NIAs.	High	AT-03
FR-46	Board page	Students must see a well formed board with a clear path of categories.	High	AT-07

FR-47	Colour changing categories	Each square on the board must show a different colour depending on the category of the question.	Low	AT-07
FR-48	Question page	When entering in a square a question and all the possible answers must be shown.	High	AT-08
FR-49	Right/wrong page	After answering a screen must inform the student if their answer was right or wrong.	High	AT-09 AT-10
FR-50	Congrats page	After finishing every game a congratulations screen must reward the student.	High	AT-11
FR-51	Ranking page	A ranking page showing the 10 fastest players and the 10 most accurate. History of the student's games too.	High	AT-12
FR-52	Backoffice	Backoffice site covering all the related operational functional requirements for members of staff.	High	AT-13

3.6.3. Non-functional requirements

This requirements will inform about system constraints.

3.6.3.1. Performance non-functional requirements

ID	Name	Description	Priority
NR-01	Browser	The app needs to be fully operable with a *HTML5* compliant browser.	High

3.6.3.2. Interoperability non-functional requirements

ID	Name	Description	Priority
NR-02	JavaScript API	The app must communicate with *Facebook* using their *API*.	High
NR-03	Responsive Design	The app must work in any browser width, from desktop to mobile devices.	High

3.7. Testing plan

This section details the testing plan that the web application must pass to be considered successful in relation with the requirements of the previous section.

3.7.1. Format

The format is very similar to the one used for requirements:

- **ID**: A unique key to be able to discern without doubt one test from the other.
- **Name**: Overview of the test.
- **Expected input**: Data or action that the test needs to run.
- **Expected output**: Result of the action in working conditions.

3.7.2. Testing plan catalog

ID	Name	Expected input	Expected output
AT-01	Login page	Student clicks on a link to the app.	Orange screen with the name of the app and a big 'Login with Facebook' button.
AT-02	Facebook auth	Student click on the 'Login with Facebook button'	The browser opens *Facebook* and ask the student to grant permissions to the app.
AT-03	Facebook auth II	Student authorises the app	Orange screen with a form to send the name and NIA.
AT-04	Facebook auth III	Student doesn't authorise the app	Go back to login page.

AT-05	Invite request	Student populates their name and NIA.	Reward text and a refresh button.
AT-06	List all boards	Student logs in successfully.	Orange screen with a top menu and the list of all available boards.
AT-07	Board	Student logs in successfully. Clicks on a board.	A fully formed board.
AT-08	Question	Student logs in successfully. Clicks on a board. Clicks on the first question.	The question, the level of difficulty and all the possible answers.
AT-09	Right answer	Student logs in successfully. Clicks on a board. Clicks on the first question. Clicks on a right answer.	A green screen rewarding the student for answering right. Updated board with a square advanced after clicking on the 'Next' button.
AT-10	Wrong answer	Student logs in successfully. Clicks on a board. Clicks on the first question. Clicks on a wrong answer.	A red screen reporting the student for a wrong answer. Updated board with non squares advanced after clicking on the 'Next' button.
AT-11	Game over	Student logs in successfully. Clicks on a board. Clicks on all the question. Clicks on all the right answers.	A green congratulations page with statistics about the game.
AT-12	Ranking	Student logs in successfully.	A page showing two rankings and the

		Clicks on the ‘ranking’ button on top.	student’s history.
AT-13	Backoffice	Member of staff logs in.	Back office with all the menus and options.

3.7.3. Matrix of testing and requirements

	AT 01	AT 02	AT 03	AT 04	AT 05	AT 06	AT 07	AT 08	AT 09	AT 10	AT 11	AT 12	AT 13
FR-01					■								
FR-02													■
FR-03													■
FR-04	■												
FR-05													■
FR-06													■
FR-07					■								
FR-08						■							
FR-09							■						
FR-10								■					
FR-11								■					
FR-12								■					
FR-13								■					
FR-14									■	■			
FR-15											■		
FR-16									■				
FR-17								■					
FR-18								■					
FR-19											■		
FR-20												■	

FR-21												
FR-22												
FR-23												
FR-24												
FR-25												
FR-26												
FR-27												
FR-28												
FR-29												
FR-30												
FR-31												
FR-32												
FR-33												
FR-34												
FR-35												
FR-36												
FR-37												
FR-38												
FR-39												
FR-40												
FR-41												
FR-42												
FR-43												

FR-44													
FR-45													
FR-46													
FR-47													
FR-48													
FR-49													
FR-50													
FR-51													
FR-52													

4. Design

This chapter shows the design decisions made not just in terms of architecture and technical matters but on user interface design as well, this is an important part, usually overlooked in traditional software projects.

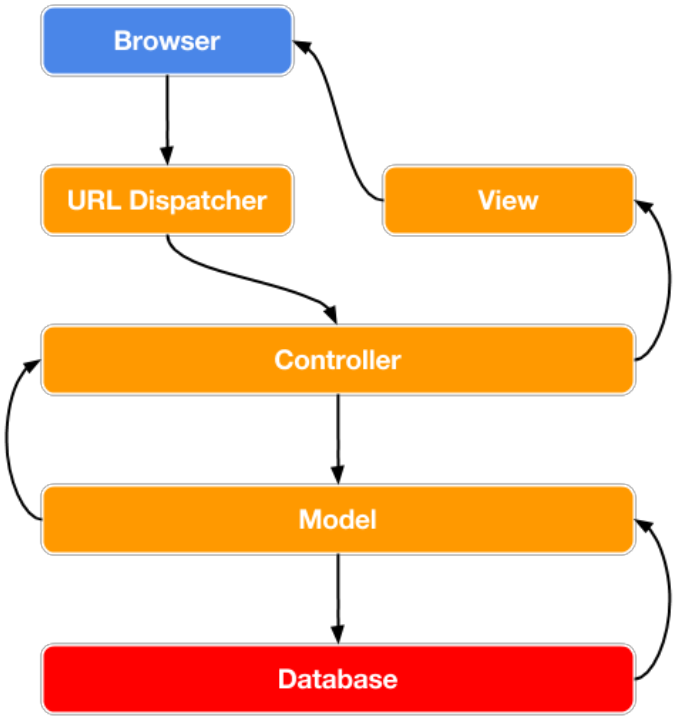
4.1. System architecture

4.1.1. Model-view-controller

Model-view-controller (MVC), was created by Trygve Reenskaug [37] in *Smalltalk-76* during the 1970s, this paradigm is very popular in web applications and basically consists in dividing the software in three parts with three different responsibilities:

- **Model**: Contains the data structures and persistence layer.
- **View**: Requests information to the *model* in order to create representations for the final user.
- **Controller**: Deals with both *model* and *view* to fulfil the application business logic.

In *Django*, the *MVC* paradigm follows the next diagram, where an *URL dispatcher* takes care of reading *HTTP* requests and routing to the correspondent *controller*.



4.1.2. Diagram of components

This is the overall architecture of components:

Views are *Student*, *Panel* and *Ranking and history*. They are responsible of what the user sees and how they interact with the application. In each *view* there are different *HTML* templates that are invoked and populated by *Django's* engine.

Controllers keep track of all the business logic, they are *Student* and *Panel*, plus an *API REST* (explained in following paragraphs). Basically they consult and act over the *models* to respond to the requests of *views*.

Models are *Student*, *Panel* and *Question*. They contain all the data and relationship between them. For the purpose of this Final Project the *models* are highly coupled with the database.

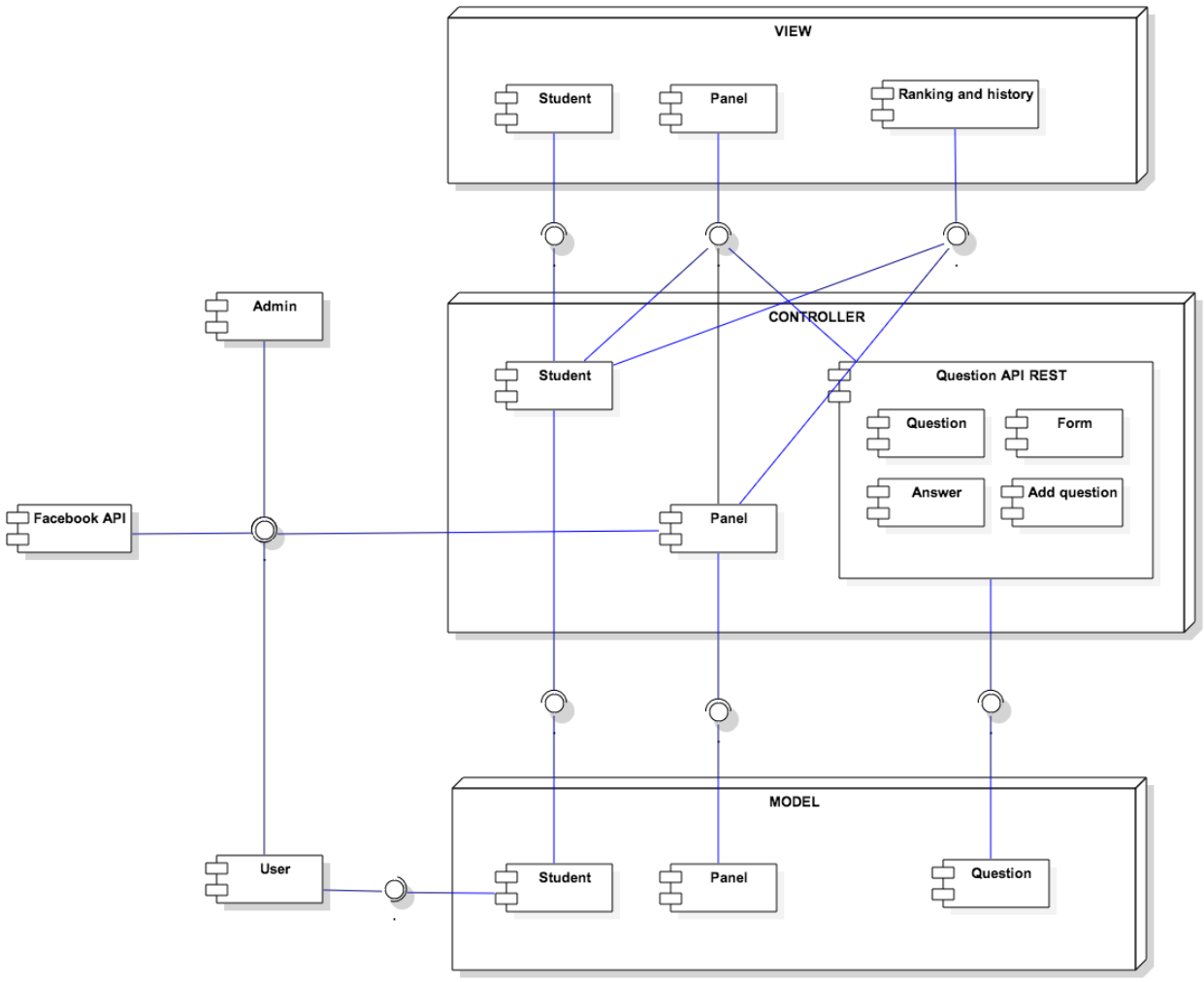
Apart of this there are three extra components on this architecture:

Facebook API is used to communicate with *Facebook* for two different actions, login (for obvious reasons) and direct messaging. That way students can send messages one to each other.

Admin is the module (*Django Suit*) that gives full access to the *models*, just for members of staff. It is a powerful way to cut down the complexity of the back office because instead of building several views and business logic pieces for each of the *models* we just need to customise the actions that members of staff are supposed to do.

User is the entity exposed by *Django* to manage user sessions and permissions, as well as the admin, a great time saver.

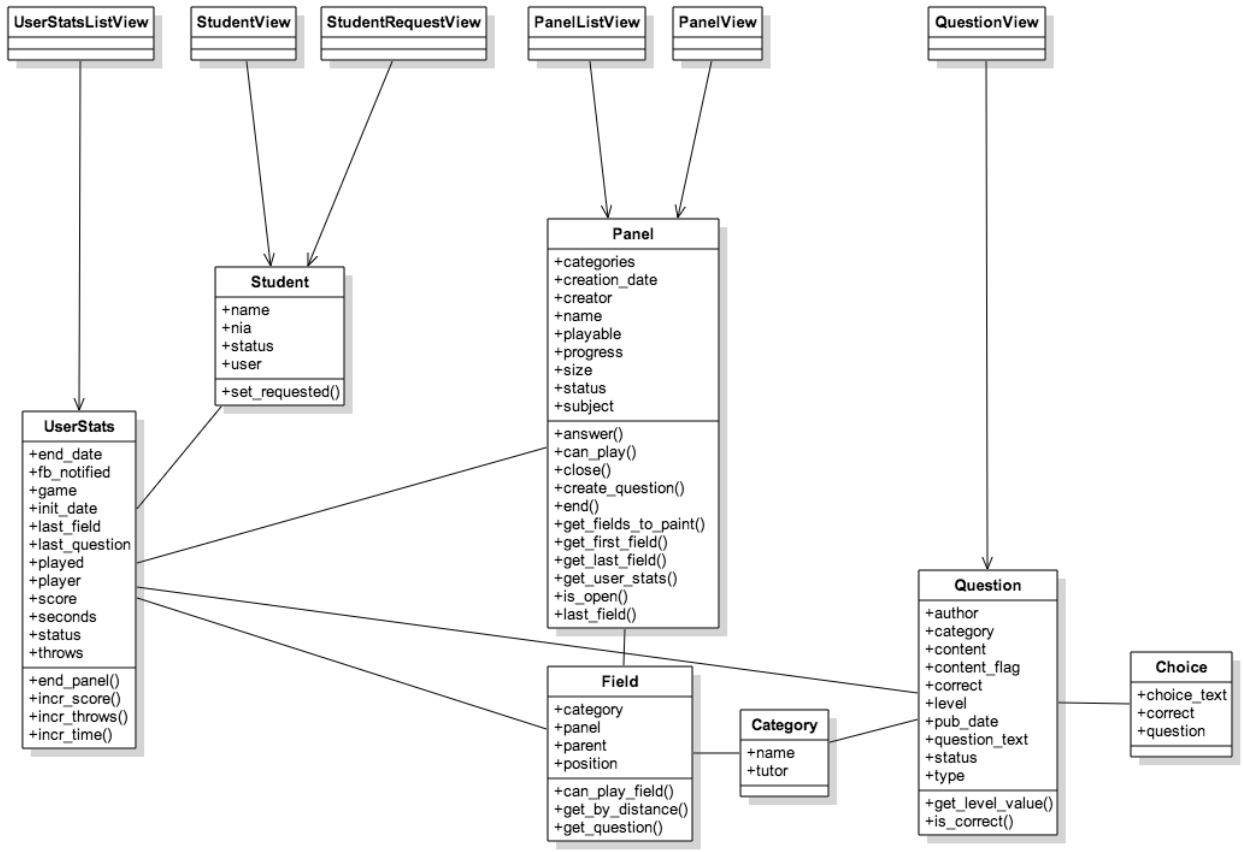
The last piece is the **Question API REST**, an internal *API* created to let the students answer questions without reloading their browser with every single petition, thanks to the asynchronous capabilities of modern browsers.


VIEW
Student
Panel
Ranking and history
Admin
CONTROLLER
Student
Question API REST
Question
Form
Answer
Add question
Facebook API
Panel
MODEL
User
Student
Panel
Question

4.2. Detailed design

This is the high definition class diagram for the most important *models*, *views* and *controllers*.

In it, all the attributes, relations and methods, which is mainly a matter of *Question*, *Panel* and *Student*.



Question contains the classes *Question*, *Choice* and *Category*, basically a category contains a series of questions, and each question contains a series of possible answers.

Apart of that also tracks the author of the question, the kind of content (if it is text or image) as well as the date of creation and status (if it is a question sent by a student can be pending or approved).

Of course, the rest of information of this class is related with the level of difficulty of the question and methods to know if a given answer is right or wrong or said level of difficulty.

Panel contains the classes *Subject*, *Panel*, *Field* and *UserStats*. A subject is managed by a tutor and can hold many panels.

Each of these panels is built with many **fields** (squares in a board), that way a board contains a fixed selection of categories but, once the player is in any field, the question that they get is completely random.

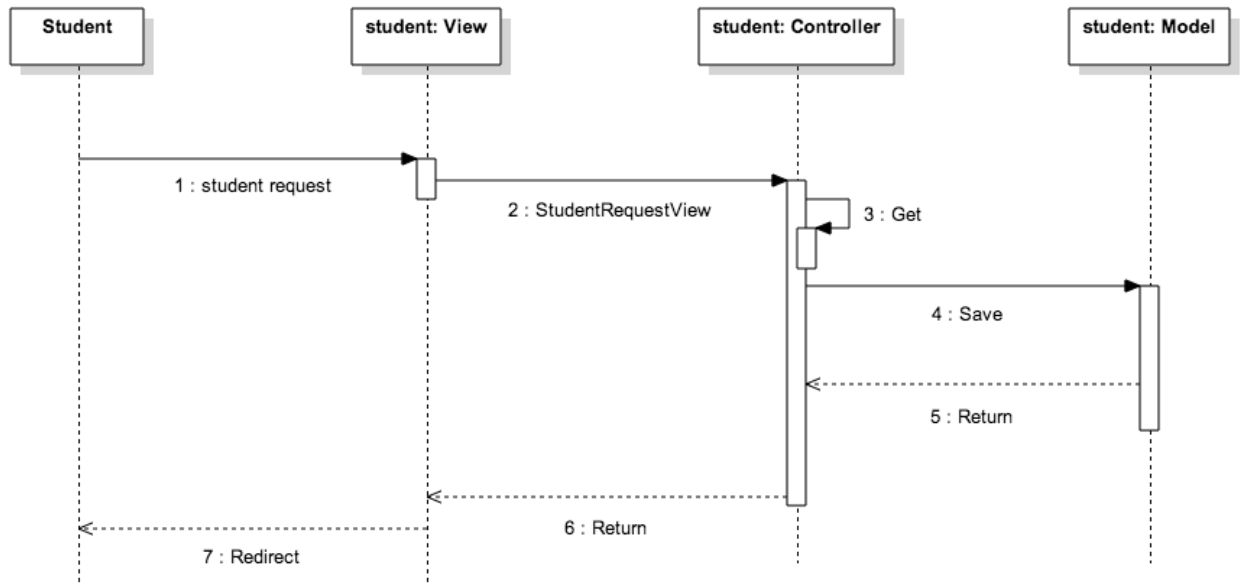
The **UserStats** class keeps track of all the games, who was the player, scores, time to completion and some internal attributes like `last_field` or `last_question` to track different sessions of the player. With all this information this class can generate rankings and history for the panel methods.

Student is the class containing all the personal information of the player, it is really simple because all the complexity of login, authorization layers and sessions is delegated on the auth and users layer that *Django* provides.

4.3. Sequence diagrams

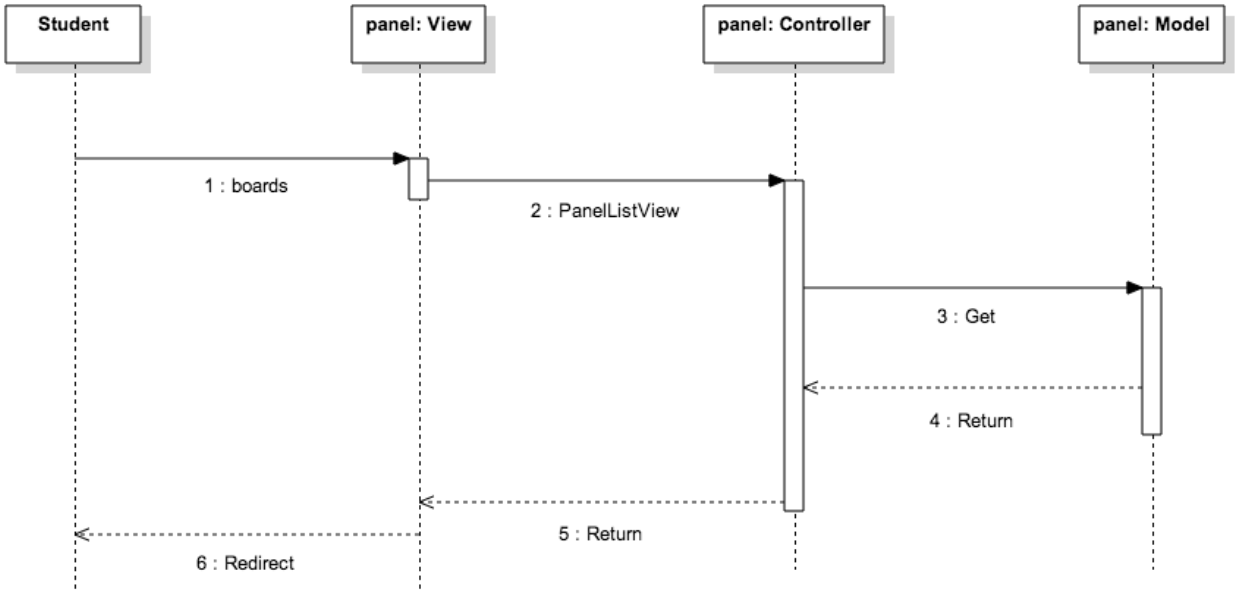
This section provides a few sequence diagrams of the most interesting parts of the web application. That way one can quickly understand what is the order of events during the normal interactions of the app.

4.3.1. Ask for an invite



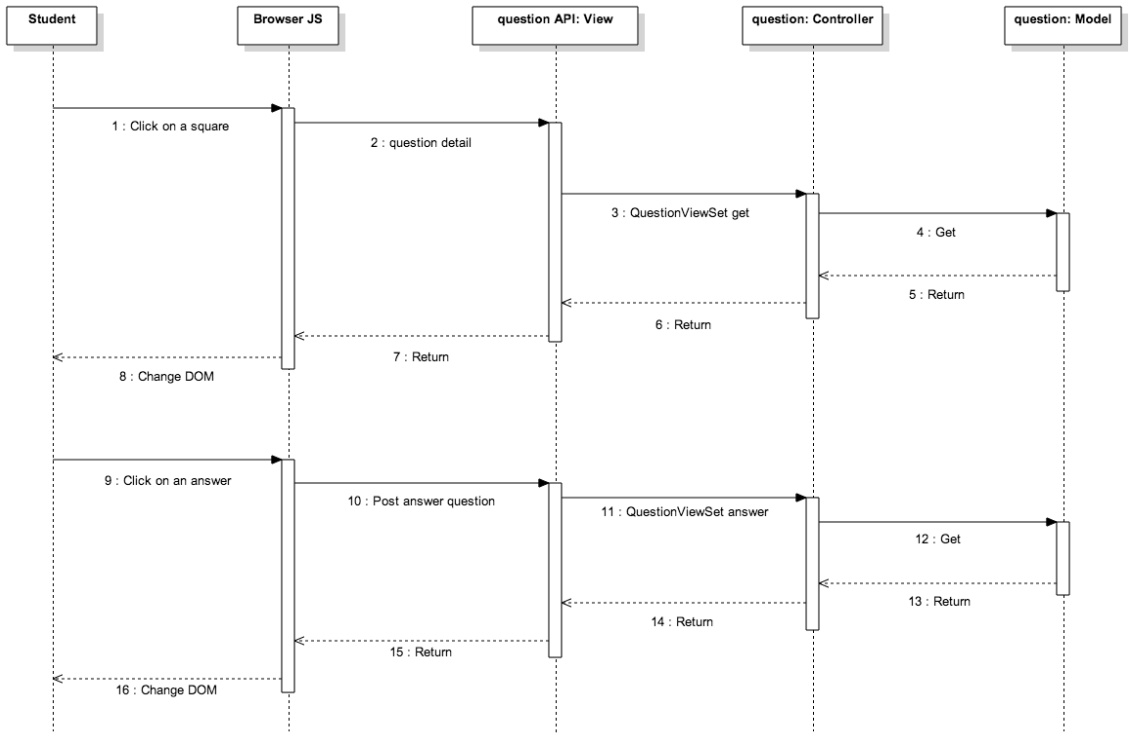
1. Student fills the form and click the submit button.
2. The *URL dispatcher* detects `student-request/` and the *view* send that submit petition to the *controller* invoking `StudentRequestView`.
3. *Controller* reads the form data through `.get` and asks the *model* to save the info with `.save`.
4. *Model* saves the information and returns the answer.
5. Different `returns` propagate to a redirect on Student's browser.

4.3.2. List all boards



1. Student clicks on Boards.
2. The *URL dispatcher* detects `boards/` and the *view* send that petition to the *controller* through `PanelListView`.
3. *Controller* asks the *model* using `.get` for the list of boards filtering all the results where `status='created'`.
4. *Model* returns that filtered list.
5. Several `returns` propagate to the Student's browser with the template `panel/list.html` populated with the contents.

4.3.3. Answer a question



1. Student clicks on a square.
2. The *JS* engine on the user's browser intercepts that call to `question/` and sends a request to the question *API REST* through `.question-detail`.
3. Controller gets the petition `QuestionViewSet` and asks the *model* for a random question of the category using `.get` and filtering `status__in=['teacher_created', 'validate']).order_by('?')`.
4. Model answers and the question and answers goes back to the *JS*, where the browser *DOM* is modified to show the new content without refreshing.
5. Student clicks on one of the answers.
6. The *JS* engine on the browser intercepts that call to `answer/` and sends a request to the question *API REST* through `.answer-question`.
7. *Controller* asks the *model* the right answer using `.answer` and check if it is the same the Student said.
8. Different `returns` propagate to the Student's browser with the new template contents to let them know whether the answer was right or wrong.

4.4. User interface

This section covers all the decisions related with the *User Interface* (UI) and front-end of the web application.

4.4.1. Design

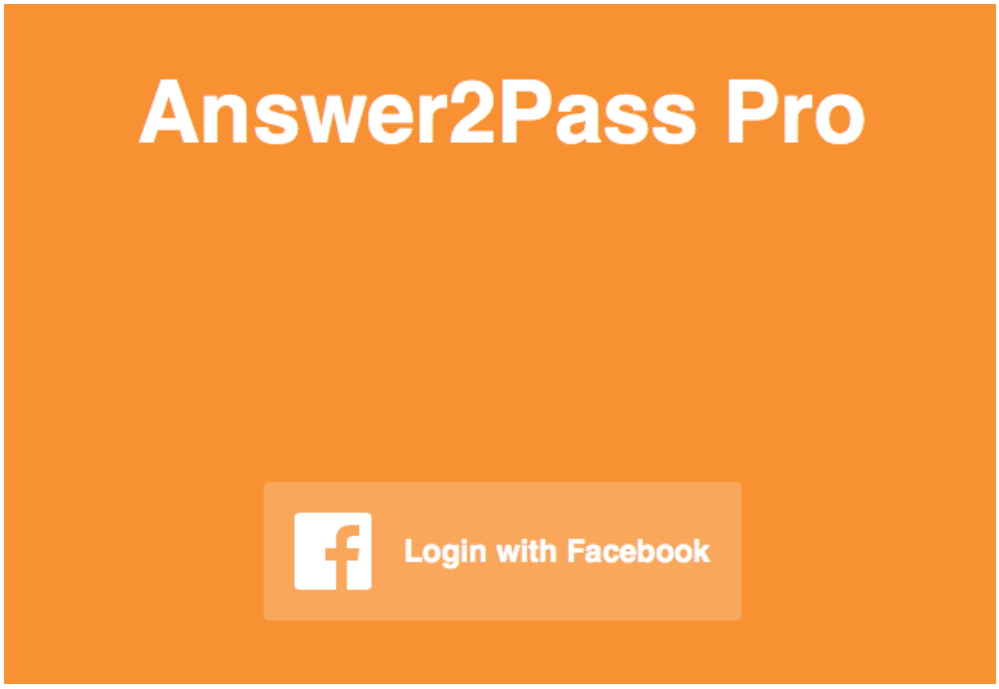
Web applications today are expected to run in every device, thats why **Answer2Pass Pro** uses *Responsive Design*, elements flow depending on the screen width at any moment and all buttons clickable are touch-friendly, big buttons and solid colours that don't rely on hover effects to convey enough affordance to make them understandable.

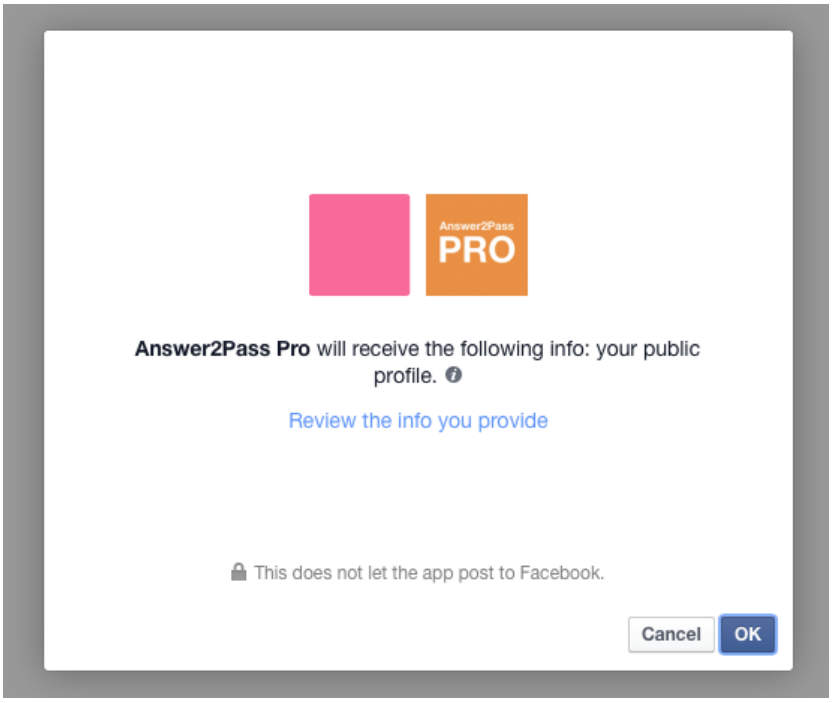
All the visual and interactive elements of the app follow the same design principles, providing consistency across the app and reducing the development costs.

A great deal of effort has been expended also in writing the right copy for every action, messages and rewards to never let the user abandoned in one-way roads. There is always a follow up action perfectly clear.

4.4.1.1. Login

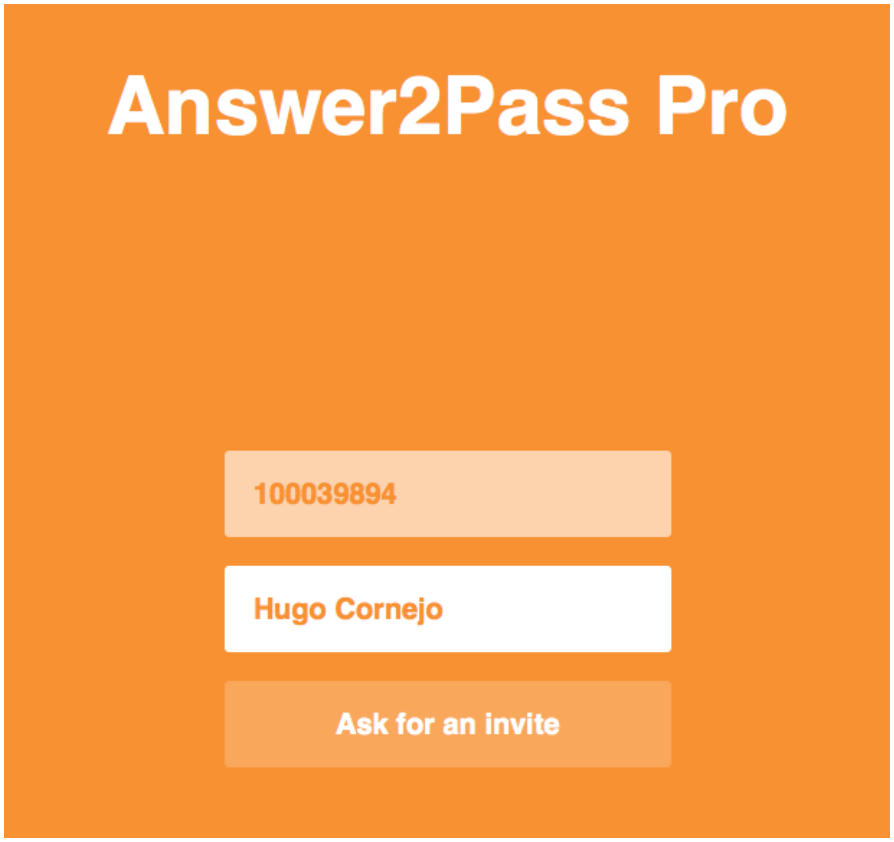
In this screen students can log in into their *Facebook* accounts in order to authorise the app to make use of their user identifiers.





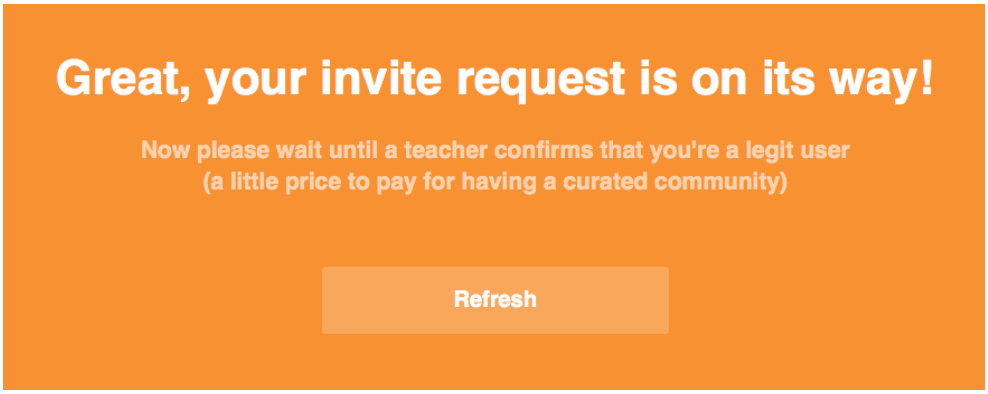
4.4.1.2. Ask for an invite

After logging in with *Facebook* students are shown with this form, they should type their NIAs and full names and click on the button.



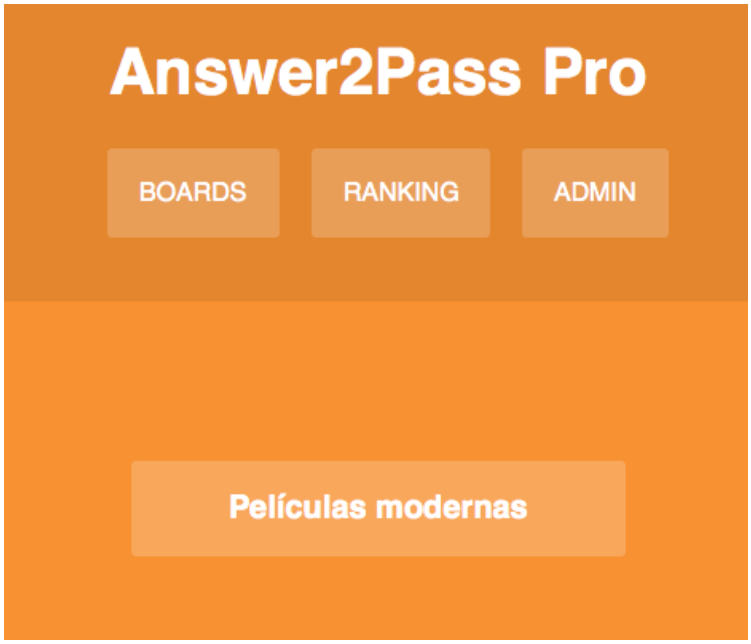
4.4.1.3. Waiting for the invite

Once they have asked for the invite they just need to wait until a member of staff accepts their request.



4.4.1.4. List all boards

After the student invite is accepted they have access to the list of all boards.

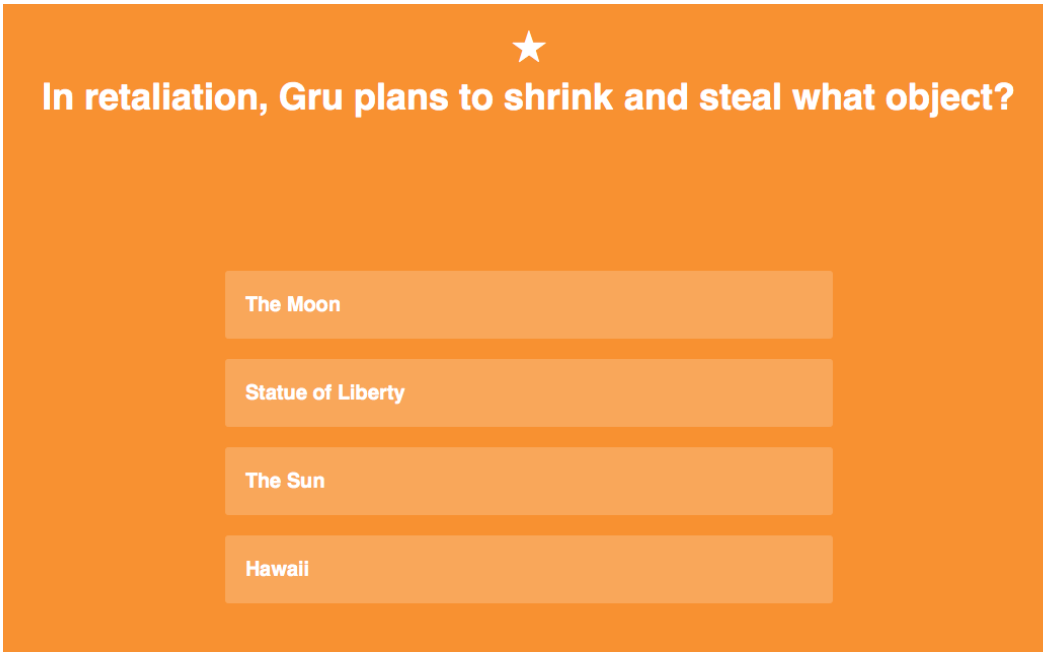


4.4.1.5. Board

Each board contains a clear path to the finish line and different colours for each square based on their categories.

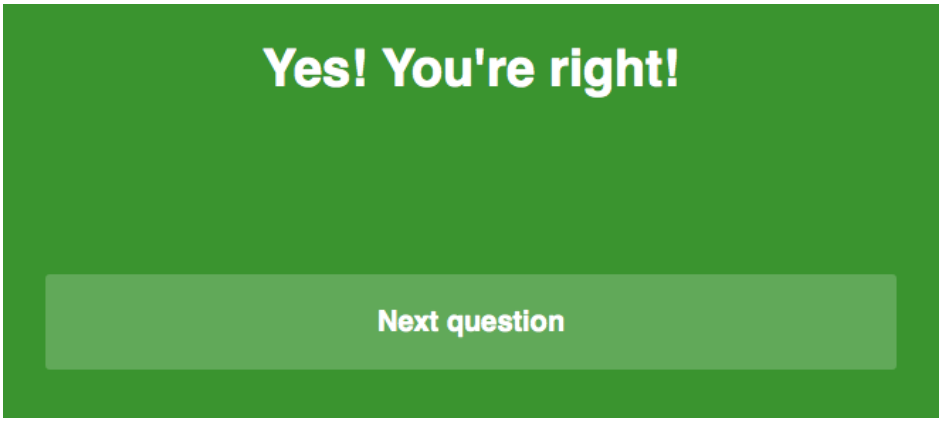
4.4.1.6. Question

Clicking on each square the student is prompted with the question and possible answers.



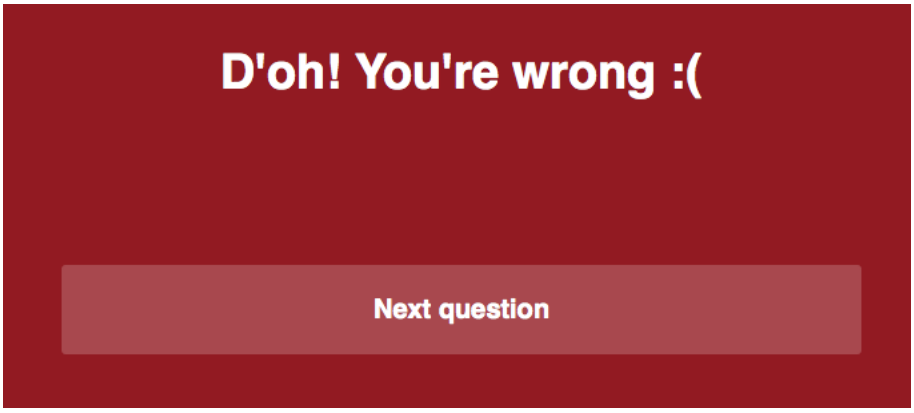
4.4.1.7. Right answer

In case that the student's answer is right the screen turns green.



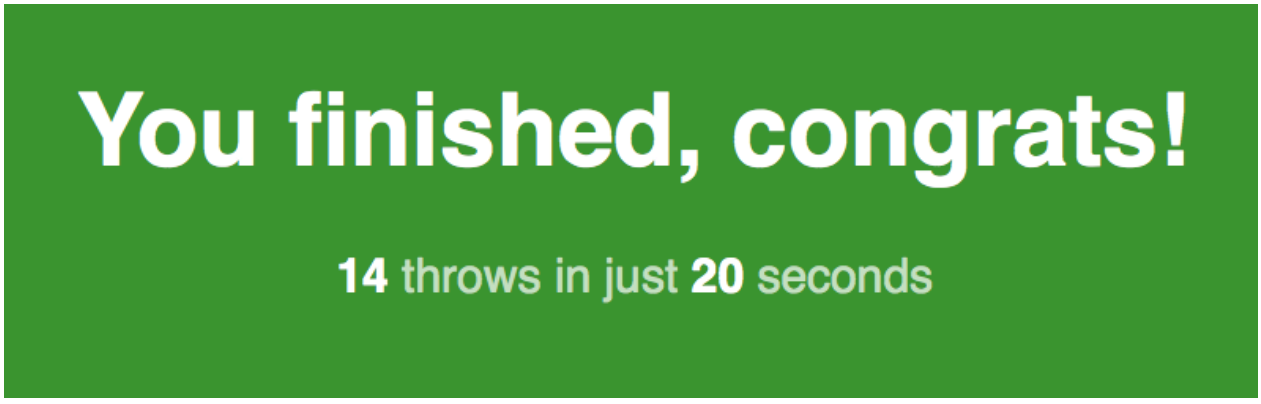
4.4.1.8. Wrong answer

In case that the student's answer is wrong the screen turns red.



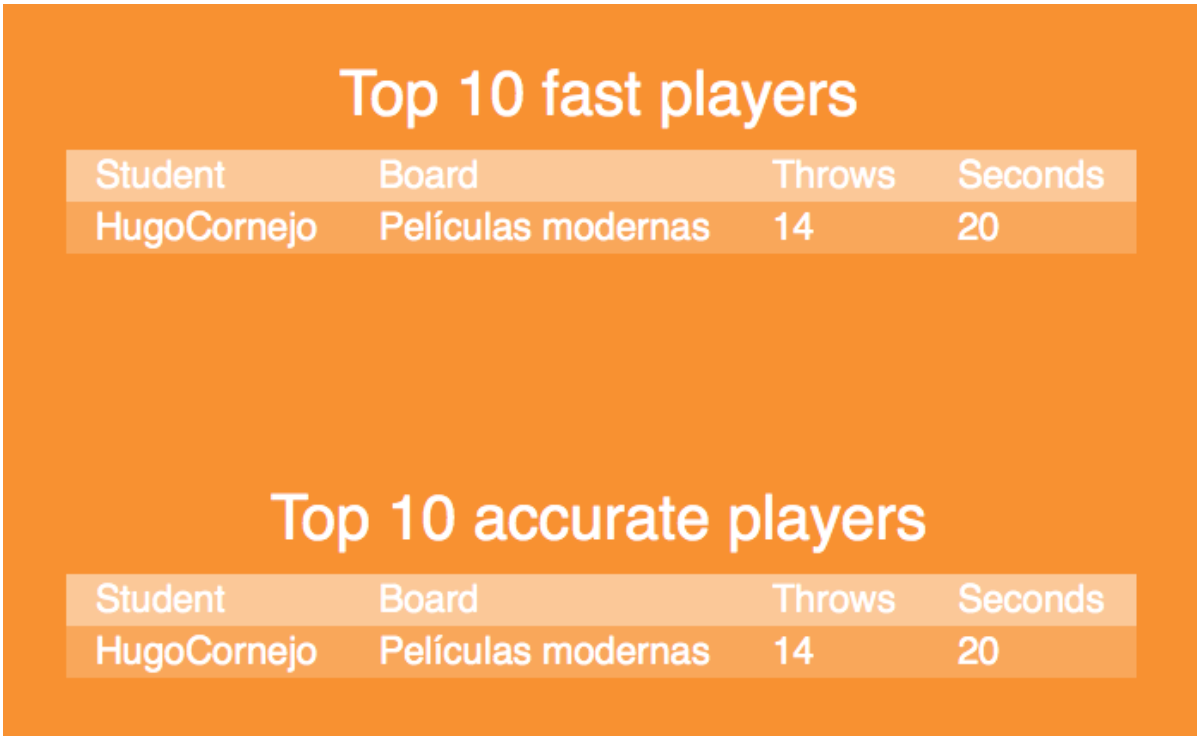
4.4.1.9. Game over

When the student reaches the finish line a congrats screen provide them with the final score.



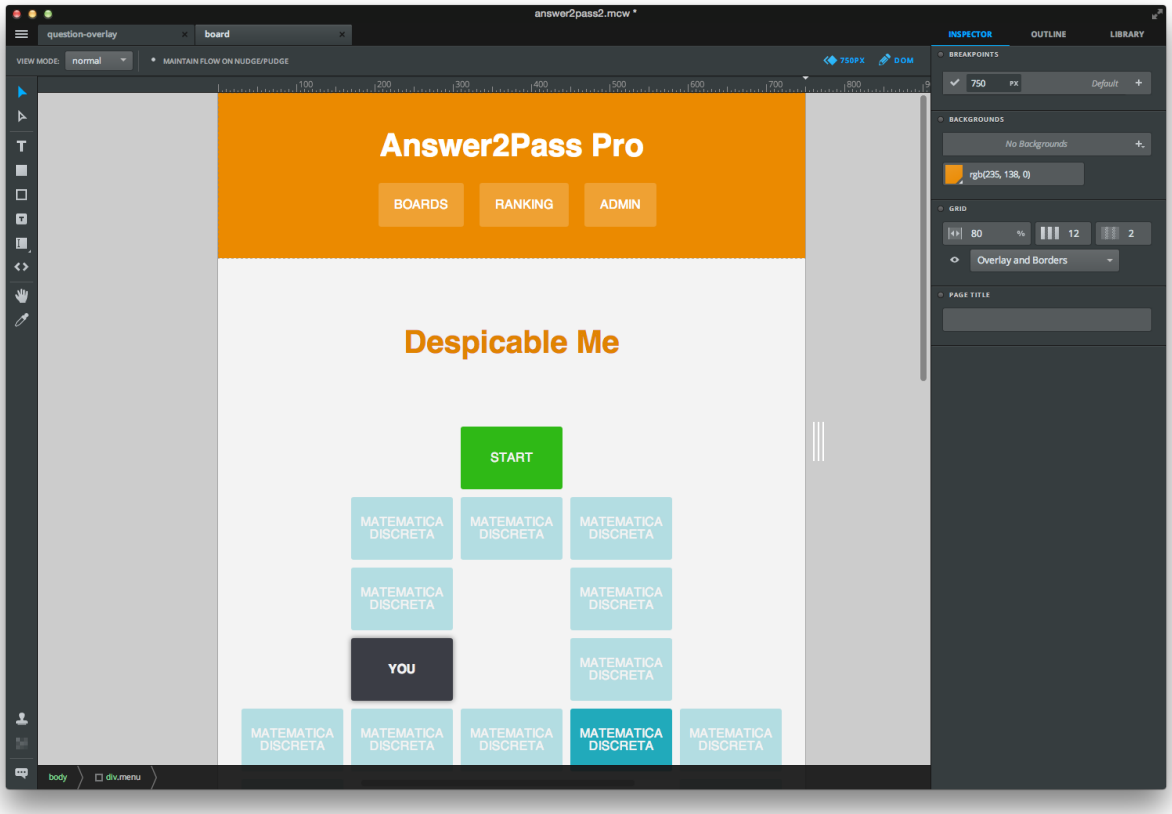
4.4.1.10. Ranking

To keep the competition amongst the students there is a ranking screen where they can compare their results with each other.



4.4.2. Front-end

Almost all the front-end was generated using *Macaw*, a web design tool which creates reasonably good *HTML* and *CSS*, easily converted then to templates compatible with *Django*.



5. Implementation

This section dives deeper in the low level details related with the implementation of the project. This is not about theoretical analysis but about real code and practice.

5.1. Libraries

One of the biggest reasons to choose *Django* for the development of this Final Project was the possibility to integrate with third parties libraries in a very easy way. Usually for matters like login and user management or *SNS* connections.

Let us explore `requirements.txt` file to see all project dependencies.

```
## DJANGO ##
django==1.7.2  # django framework, our core.

# https://github.com/gotcha/ipdb
ipdb==0.8 # debug

# http://django-social-auth.readthedocs.org/en/latest/
python-oauth2==0.7.0
django-social-auth==0.7.28

# http://djangojs.readthedocs.org/en/latest/
django.js==0.8.1

# http://www.django-rest-framework.org/
djangorestframework==3.0.5

# http://django-dynamic-fixture.readthedocs.org/en/latest/index.html
django-dynamic-fixture==1.8.1

# http://django-suit.readthedocs.org/en/develop/
django-suit==0.2.12

# http://django-facebook.readthedocs.org/
django-facebook==6.0.2

# http://pillow.readthedocs.org/
pillow==2.7.0
```

Now we can analyse in detail the most important of these requirements and dependencies.

5.1.1. Django

This is the core frameworks that provides all the *MVC* capabilities and a big suite of tools to run the code in a server, transfer initial data to the database, deal with assets and other everyday tasks of web development.

5.1.2. Django Social Auth

A simple library to use *Facebook* platform as an authentication provider. It supports dozens of platforms which would make really easy to extend the login options to other providers: e.g. *Twitter*, *LinkedIn*, *Flickr*, *Dropbox*, etc.

5.1.3. django.js

Library full of tools for *JavaScript* development. Used for the question modal window and the asynchronous interactive parts.

5.1.4. Django REST framework

This toolkit allows the building of the internal *API* that students browsers call straight from *JavaScript*, without going through the *Python* server. This provides a cleaner architecture and improves the performance on the server side.

5.1.5. Django Suit

This library provides an easier to use alternative theme for the *Django* administration interface application.

The *Django* admin app is what makes that all the members of staff related actions of this Final Project are covered without barely writing a single line of code. *Django* admin app takes models and some imperative declarations and builds automatically all the views, forms, controls and validations needed to operate these structures.

Without this little help the Final Project would have required probably two or three times more resources to be completed.

5.1.6. Django Facebook

This is a library to access students' *Facebook Graph*, something required to let the application publish in their *Facebook* wall on their behalf and send *DM* to their friends.

5.2. Testing results

In order to consider the application ready to use the testing plan is run manually twice. Happily, the software passes successfully all the testing plan previously defined.

This table summarises that:

ID	Result
AT-01	✓ PASSED
AT-02	✓ PASSED
AT-03	✓ PASSED
AT-04	✓ PASSED
AT-05	✓ PASSED
AT-06	✓ PASSED
AT-07	✓ PASSED
AT-08	✓ PASSED
AT-09	✓ PASSED
AT-10	✓ PASSED
AT-11	✓ PASSED
AT-12	✓ PASSED
AT-13	✓ PASSED

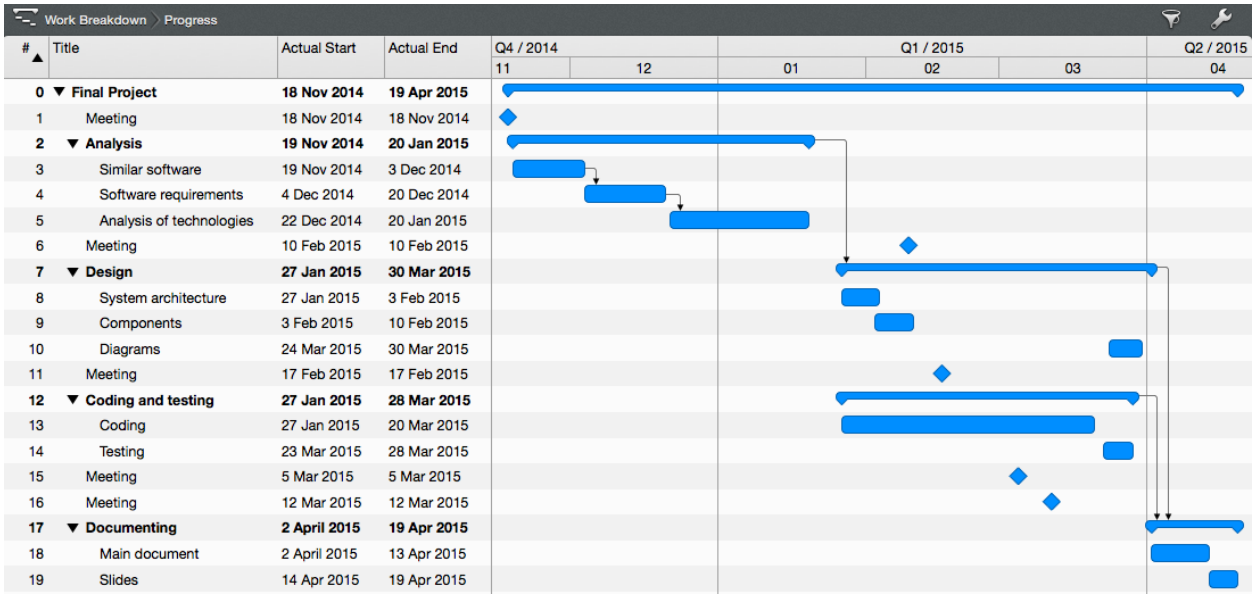
6. Project planning

In all honesty, the planning of this Final Project was particularly chaotic. During the first two months the author tried to build **Answer2Pass Pro** on top of the original **Answer2Pass**, the result was a disaster due to the lack of *Java* skills on the author side.

With that initial waste of time the author seriously considered giving up on the endeavour but thanks to the directors he found motivation again and in January he started from scratch a new project on *Django*, aiming to port the original **Answer2Pass** plus all the new required features.

From that moment on the planning worked really well, with not a single delay on deliveries or meetings.

See the following gantt chart for more details.



#	Title	Actual Start	Actual End
0	**Final Project**	**18 Nov 2014**	**19 Apr 2015**
1	Meeting	18 Nov 2014	18 Nov 2014
2	**Analysis**	**19 Nov 2014**	**20 Jan 2015**
3	Similar software	19 Nov 2014	3 Dec 2014
4	Software requirements	4 Dec 2014	20 Dec 2014
5	Analysis of technologies	22 Dec 2014	20 Jan 2015
6	Meeting	10 Feb 2015	10 Feb 2015
7	**Design**	**27 Jan 2015**	**30 Mar 2015**
8	System architecture	27 Jan 2015	3 Feb 2015
9	Components	3 Feb 2015	10 Feb 2015
10	Diagrams	24 Mar 2015	30 Mar 2015
11	Meeting	17 Feb 2015	17 Feb 2015
12	**Coding and testing**	**27 Jan 2015**	**28 Mar 2015**
13	Coding	27 Jan 2015	20 Mar 2015
14	Testing	23 Mar 2015	28 Mar 2015
15	Meeting	5 Mar 2015	5 Mar 2015
16	Meeting	12 Mar 2015	12 Mar 2015
17	**Documenting**	**2 April 2015**	**19 Apr 2015**
18	Main document	2 April 2015	13 Apr 2015
19	Slides	14 Apr 2015	19 Apr 2015

6.1. Technical resources used

This section describes all the technical resources used (just by the author) during the creation of this Final Project.

6.1.1. Hardware

- Macbook Air 13” mid-2011
- Display Dell 27” U2713HM
- iPhone 5
- iPad Air 2

6.1.2. Software

- Mac OS X
- Google Chrome
- PyCharm
- Macaw
- Sublime Text
- Sketch
- Star UML
- Google Docs
- Nitrous
- Dropbox

6.2. Financial analysis

This section analyses all the costs related to the development of **Answer2Pass Pro** in terms of developer time, but also hardware, software and services.

All the figures include VAT.

6.2.1. Personnel costs

The total time invested in developing this app has been 264 hours (2 hours a day during 12 weeks, 12 hours on weekends). Considering a rate of 68 euros per hour, real author's freelance rate for 2015.

Hours	Rate	Cost
264	68 €/h	**21,721.92 €**

6.2.2. Hardware costs

Item	Life cycle	Time used	Full price	Cost
MacBook Air	48 months	3 months	1,500.00 €	93.75 €
Display Dell	48 months	3 months	573.88 €	35.87 €
iPhone 5	24 months	3 months	552.95 €	69.12 €
iPad Air 2	36 months	3 months	809.00 €	67.42 €
				266.16 €

6.2.3. Software costs

Item	Life cycle	Time used	Licence price	Cost
PyCharm	24 months	3 months	214.00 €	26.75 €
Macaw	24 months	3 months	164.36 €	20.55 €
Sketch	24 months	3 months	90.90 €	11.36 €
Nitrous	Monthly	2 months	18.36 €	36.72 €
Dropbox	Annual	3 months	90.90 €	22.73 €
				118.11 €

6.2.4. Operative costs

Item	Monthly rate	Time used	Cost
Desk in a coworking space	410.00 €	3 months	**1230.00 €**

6.2.5. Total costs

Department	Cost
Personnel	17,952.00 €
Hardware	266.16 €
Software	118.11 €
Operative costs	1230.00 €
	23,336.19 €
Before VAT	**19,286.11 €**

6.2.6. Total budget after risk and profit

Concept	Cost
Base	23,336.19 €
Risk (15%)	3,500.43 €
Profit (25%)	5,834.05 €
	32,670.67 €
Before VAT	**27,000.55 €**

7. Conclusions

This section explores the conclusions after the development of **Answer2Pass Pro** as well as possible future lines of work.

7.1. Project conclusions

The first conclusion of the Final Project is that the software has been correctly developed on time, everything works and the idea seems financially viable.

Said software was an elearning platform in which teachers can create boards of questions (quizzes) for their students to play, testing their knowledge answering questions, learning new topics and even sending their own questions to the platform or other fellow students.

On the technical side, another conclusion is that *Django* is a great platform to create web products, relieving the developer from much of the complexity of database access, auth systems and administration back offices. It was great too to have access to a huge diversity of libraries to extend functionalities, e.g. *Facebook*.

7.2. Personal conclusions

To understand my personal conclusions first I need to tell you a bit about me.

I joined the UC3M on 2002 when I was 17 years old. By 2006 I passed all the subjects, since then I have been intermittently trying to put a full stop to the university years presenting a proper Final Project.

Well, it has been 9 years, now I am 30. In this time I worked as a developer for a few years and then decided to study Interaction Design. Today I am working as a

Lead Designer in an amazing fintech startup and my developer years are gone for good.

So this Final Project for me was not at all about software, it was about closure. I enjoyed knowing that my rusted programming skills are still enough to pull a software piece from scratch and on time but, most of all, I enjoyed knowing that I was at last closing my university endeavour.

7.3. Future works

There are lots of possible areas to explore in a future version of **Answer2Pass Pro**, all of them would make the different business models explored even more interesting and viable.

- **Integrate more login providers**. Thanks to the employed architecture it would be really easy to add other sign in providers apart of *Facebook* to the pool. Integrating *Twitter* or *LinkedIn* will extend the options for students, making the experience much easier for them.
- **Delayed sign in**. One interesting improvement would be to create a delayed sign in process in which students can play a board and only sign in once they finish. Ask for permissions in *Facebook* is usually a high toll when the user has not had the opportunity to see exactly what is they are applying for.
- **Boards with different shapes**. Apart from the three different board sizes created in this Final Project it would be great to create an algorithm of random board generation, that way each game would be completely new.
- **New kinds of questions**. Right now **Answer2Pass Pro** offers true/false questions and multiple choice, but it would be feasible to extend that to other kind of games as 'fill in gaps', crosswords, etc.
- **Multiplayer on the same board**. A big improvement (inspired in the competitor *Kahoot!*) would be to allow members of staff to create special boards to be played during class hours by all the students on the same board, in real time. Thanks to the *Responsive Design* students could play from their

mobile devices in a big race to see who is the fastest on finishing the board or the one who answers correctly more questions during a period of time.

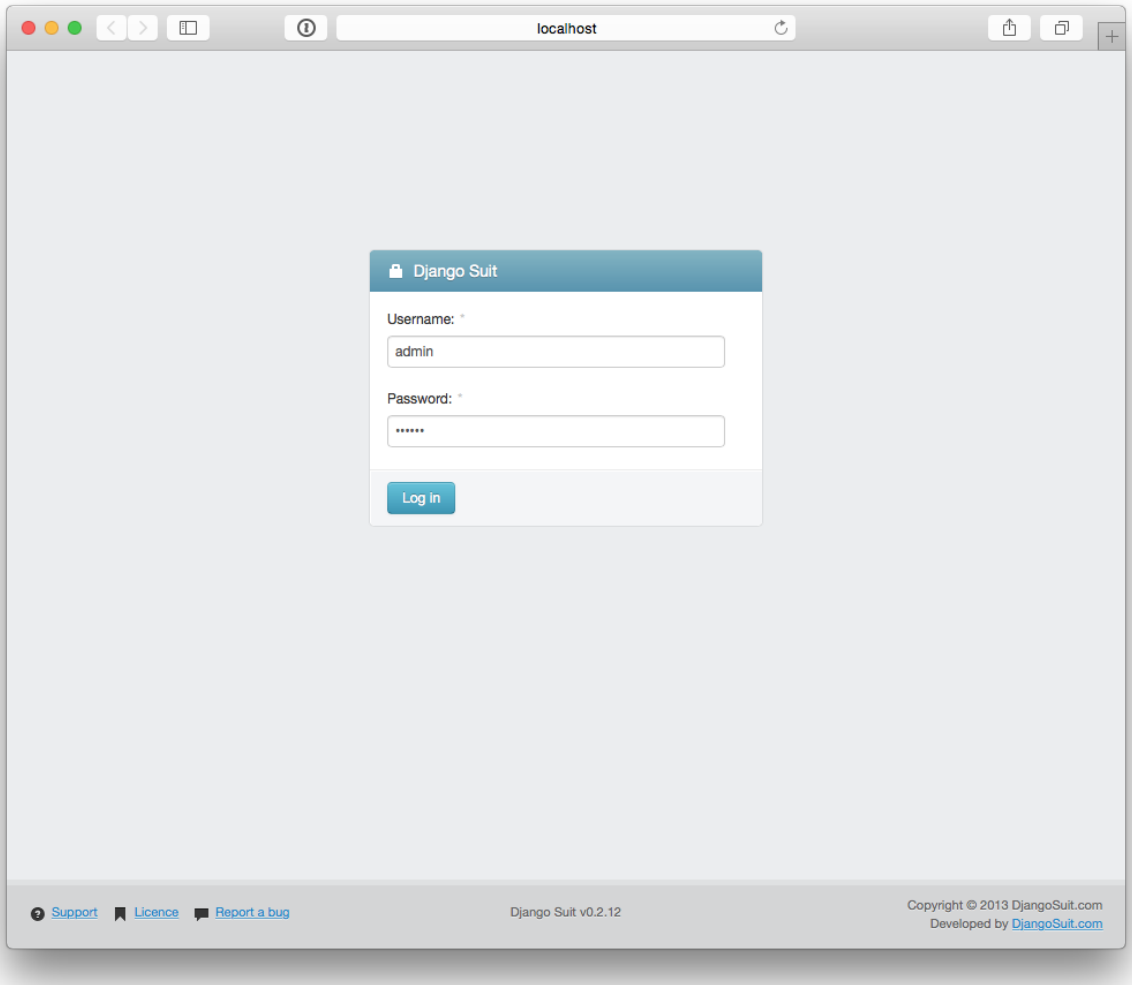
8. User manual for members of staff

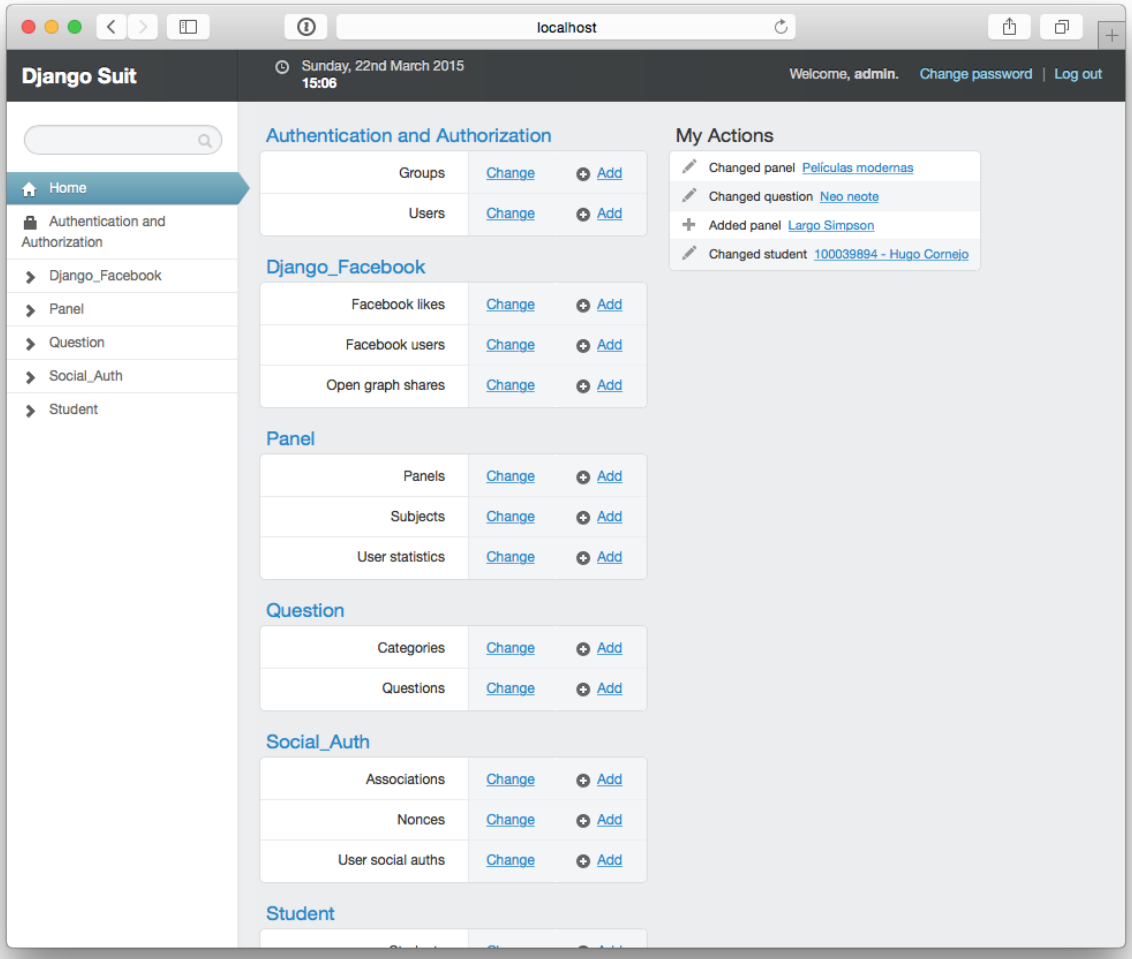
Due to the nature of the project there is an irreducible complexity on the members of staff (coordinators and administrator) side of the app. This complexity can be tackled with a proper user manual, built to answer specific questions in a how-to way.

In the other hand, the student side of the app is designed to be used without any manual or guide apart from the very design affordances and texts.

8.1. How to log in as a member of staff?

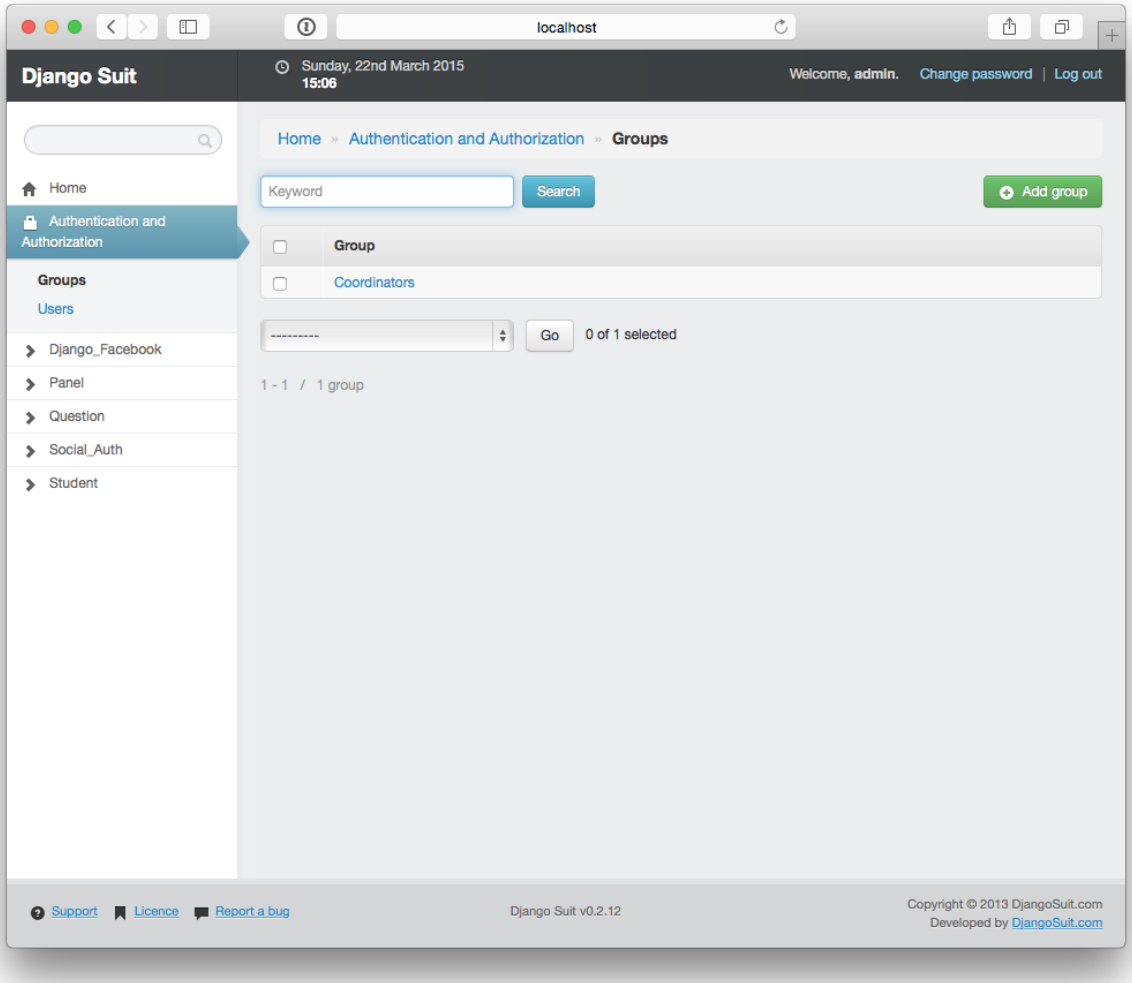
1. Open a browser and go to /admin
2. Enter your credentials.
3. Click the Login button.

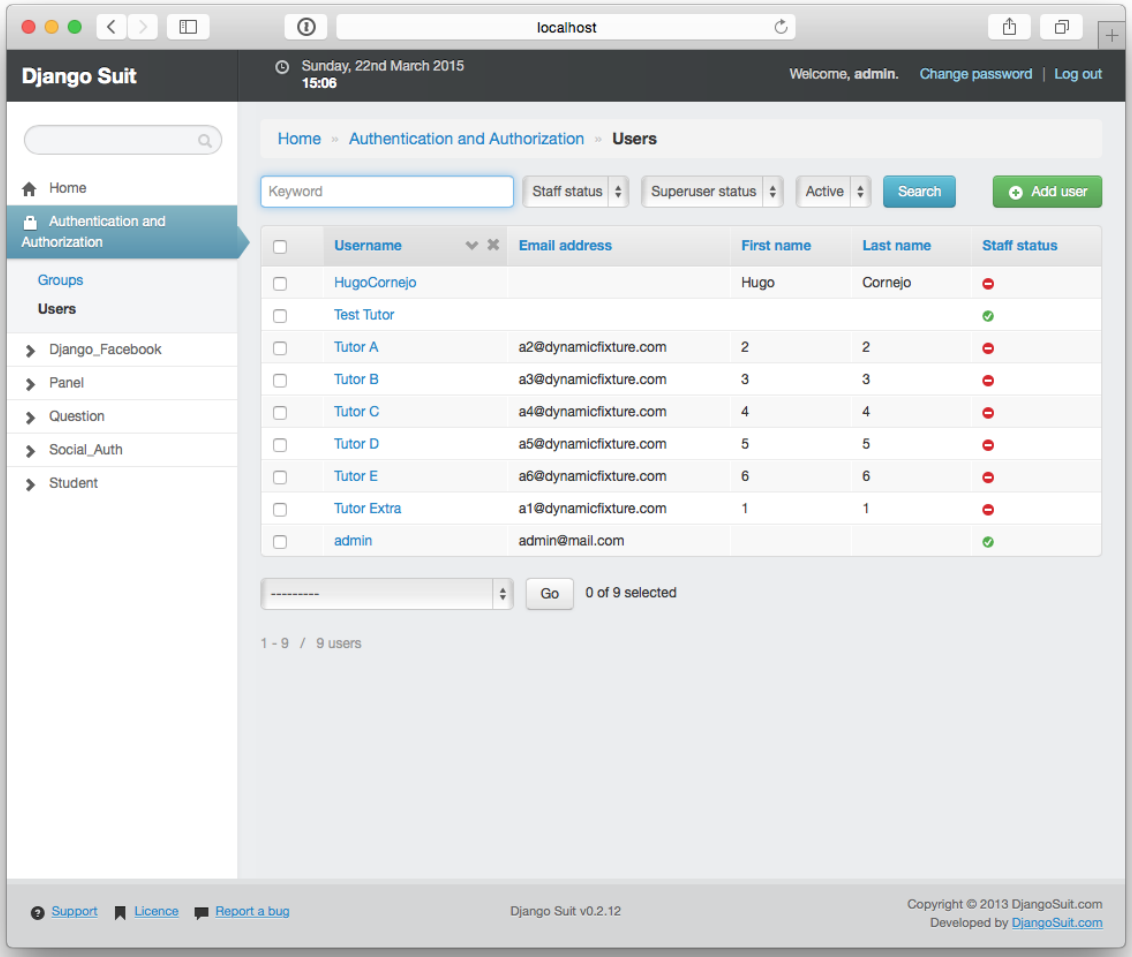



localhost
Django Suit
Sunday, 22nd March 2015
15:06
Welcome, admin.
Change password
Log out
Home
Authentication and Authorization
Django_Facebook
Panel
Question
Social_Auth
Student
Authentication and Authorization
Groups Change Add
Users Change Add
Django_Facebook
Facebook likes Change Add
Facebook users Change Add
Open graph shares Change Add
Panel
Panels Change Add
Subjects Change Add
User statistics Change Add
Question
Categories Change Add
Questions Change Add
Social_Auth
Associations Change Add
Nonces Change Add
User social auths Change Add
Student
My Actions
Changed panel Películas modernas
Changed question Neo neote
Added panel Largo Simpson
Changed student 100039894 - Hugo Cornejo

8.2. How to manage other members of staff?

1. Go to Authentication and authorization link in the sidebar.
2. Dive in the Users section.
3. Click on a user to change their properties or click in the Add user button to create a new one.




localhost
Django Suit
Sunday, 22nd March 2015
15:06
Welcome, admin.
Change password
Log out
Home
Authentication and Authorization
Groups
Users
Django_Facebook
Panel
Question
Social_Auth
Student
Home » Authentication and Authorization » Users
Keyword
Staff status
Superuser status
Active
Search
Add user
Username
Email address
First name
Last name
Staff status
HugoCornejo
Hugo
Cornejo
Test Tutor
Tutor A
a2@dynamicfixture.com
2
2
Tutor B
a3@dynamicfixture.com
3
3
Tutor C
a4@dynamicfixture.com
4
4
Tutor D
a5@dynamicfixture.com
5
5
Tutor E
a6@dynamicfixture.com
6
6
Tutor Extra
a1@dynamicfixture.com
1
1
admin
admin@mail.com

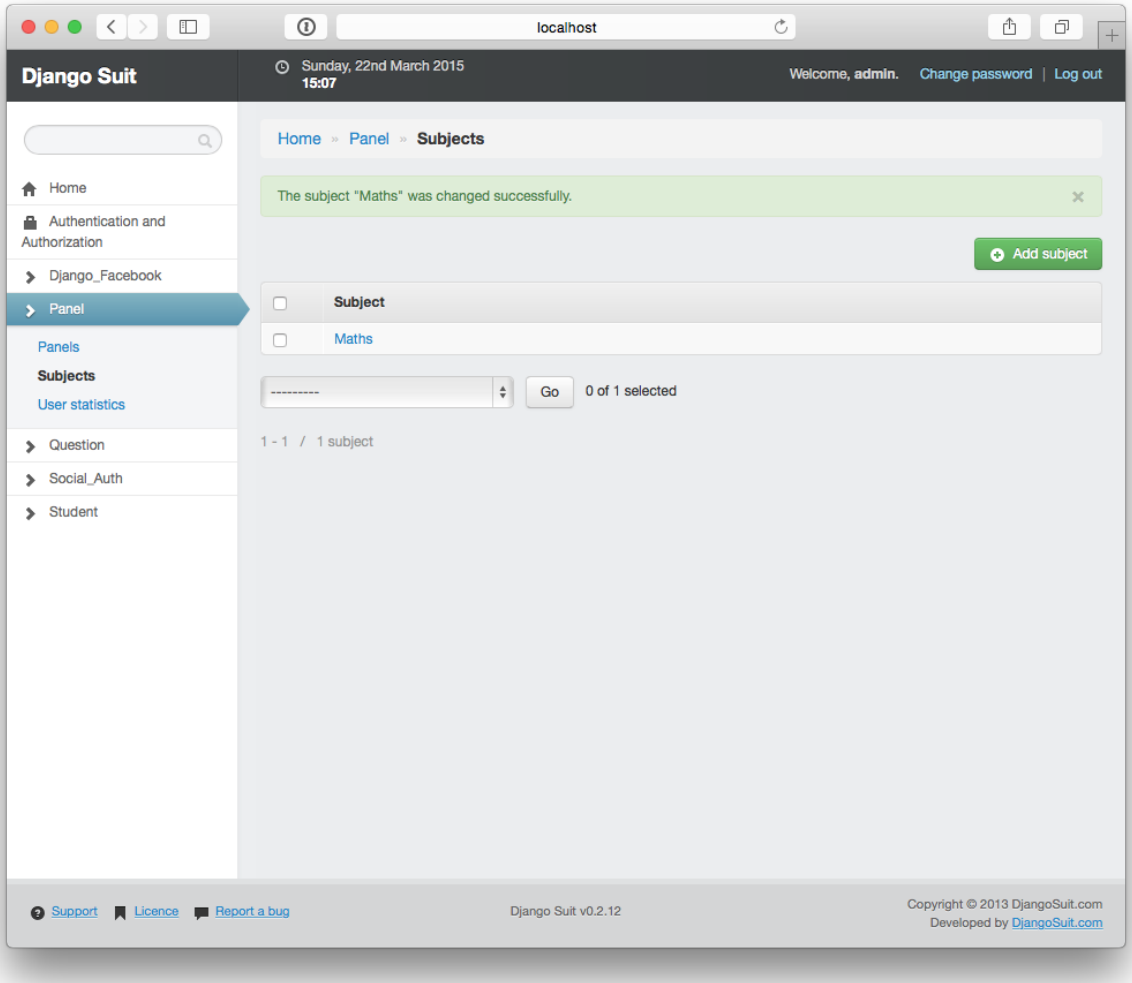
Go
0 of 9 selected
1 - 9 / 9 users
Support
Licence
Report a bug
Django Suit v0.2.12
Copyright © 2013 DjangoSuit.com
Developed by DjangoSuit.com

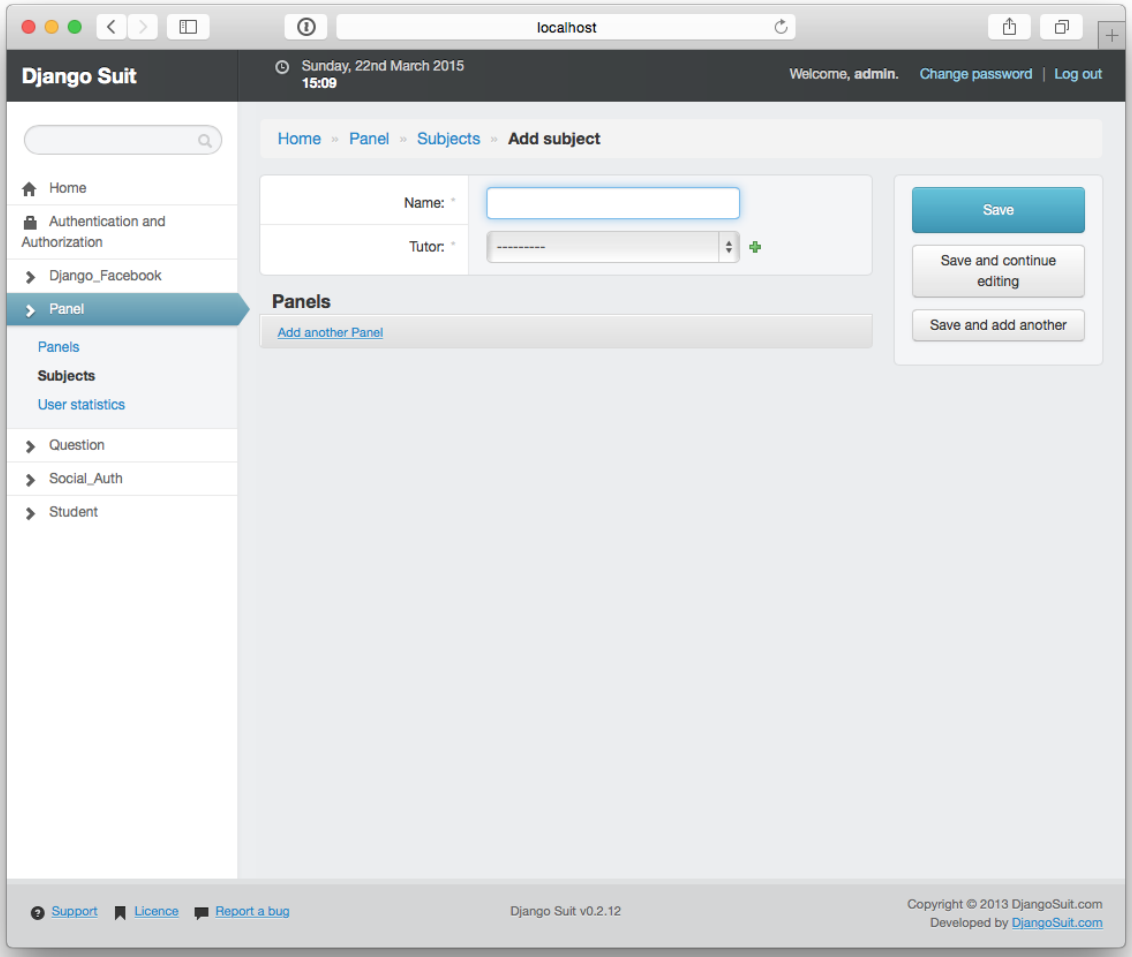
8.3. How to create a new board?

1. Go to the Panel section.
2. Define a name, a creator and a subject.
3. Choose the categories, including or not the Extra squares.
4. Pick a size (small, medium or large).
5. Define if the progress is standard (the player progresses more answering difficult questions) or simple (every right answer is one square progress).
6. Choose for how long can the board be played, or even if it can only be played once by each student.
7. Click Save to create the board.

8.4. How to create a new subject?

1. Go to the Panel > Subject section.
2. Click on the Add subject button.
3. Define a name and choose a tutor responsible for it.
4. Click Save to keep the changes.

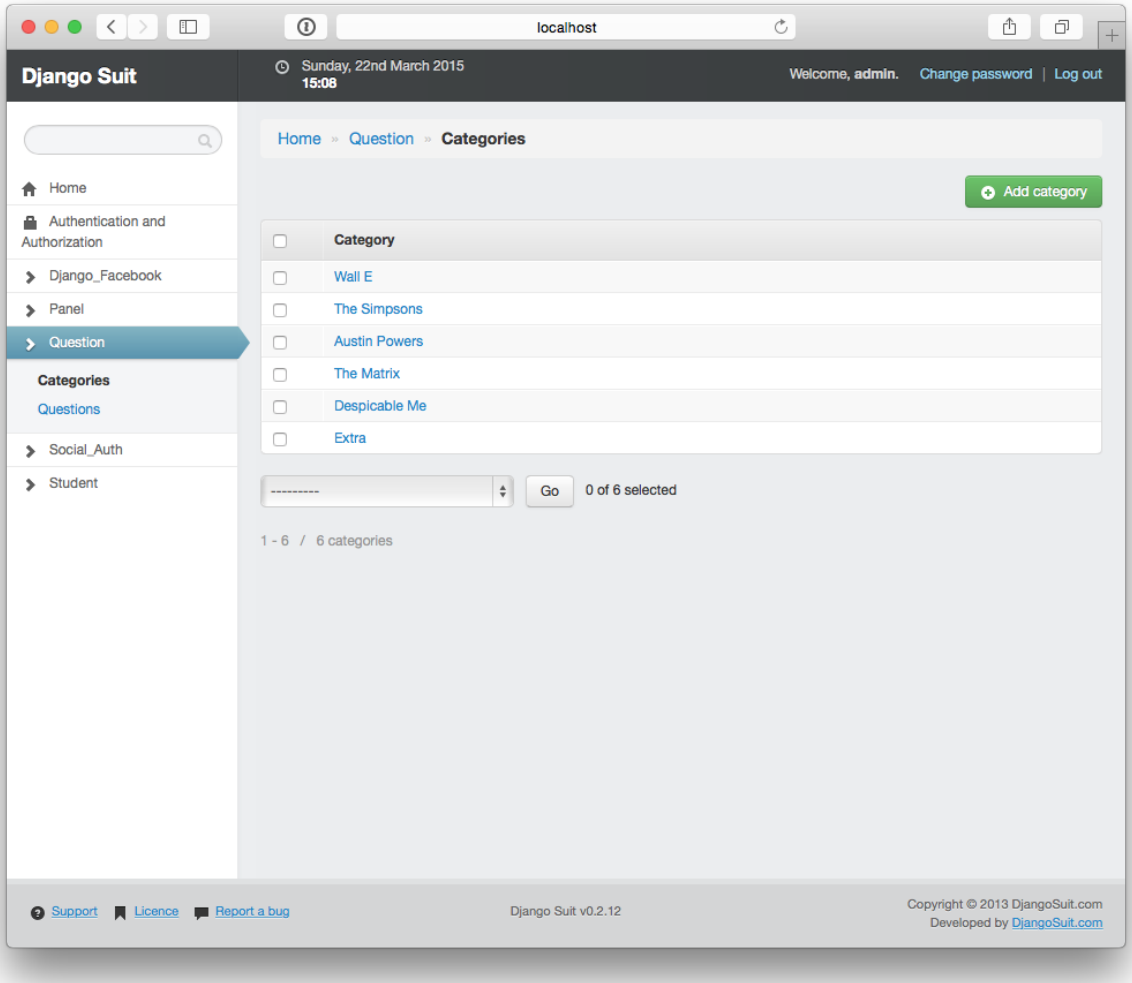


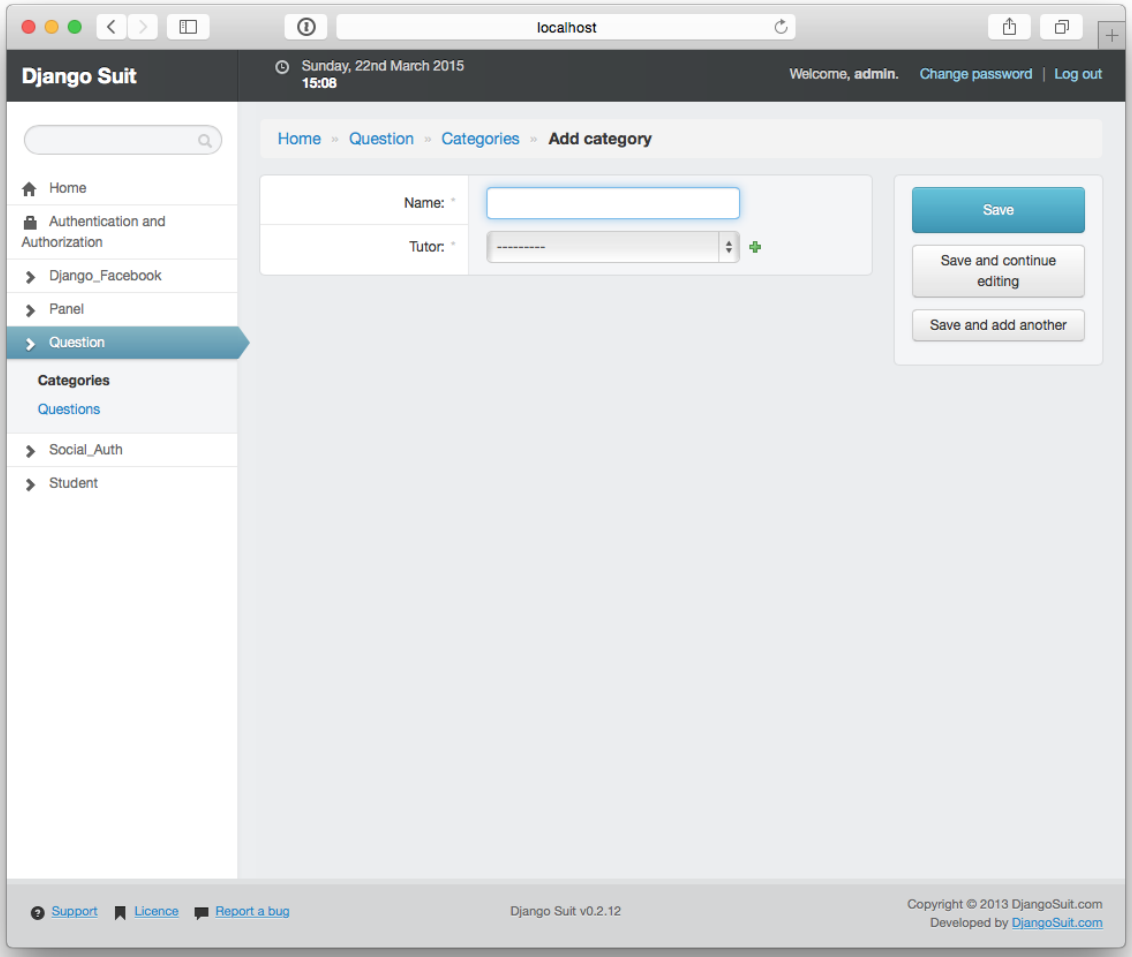

localhost
Django Suit
Sunday, 22nd March 2015
15:09
Welcome, admin.
Change password
Log out
Home
Authentication and Authorization
Django_Facebook
Panel
Panels
Subjects
User statistics
Question
Social_Auth
Student
Home » Panel » Subjects » Add subject
Name: *
Tutor: *

Save
Save and continue editing
Save and add another
Panels
Add another Panel
Support
Licence
Report a bug
Django Suit v0.2.12
Copyright © 2013 DjangoSuit.com
Developed by DjangoSuit.com

8.5. How to create a new category?

1. Go to the Question > Categories section.
2. Click on the Add category button.
3. Define a name and choose a tutor responsible for it.
4. Click Save to keep the changes.

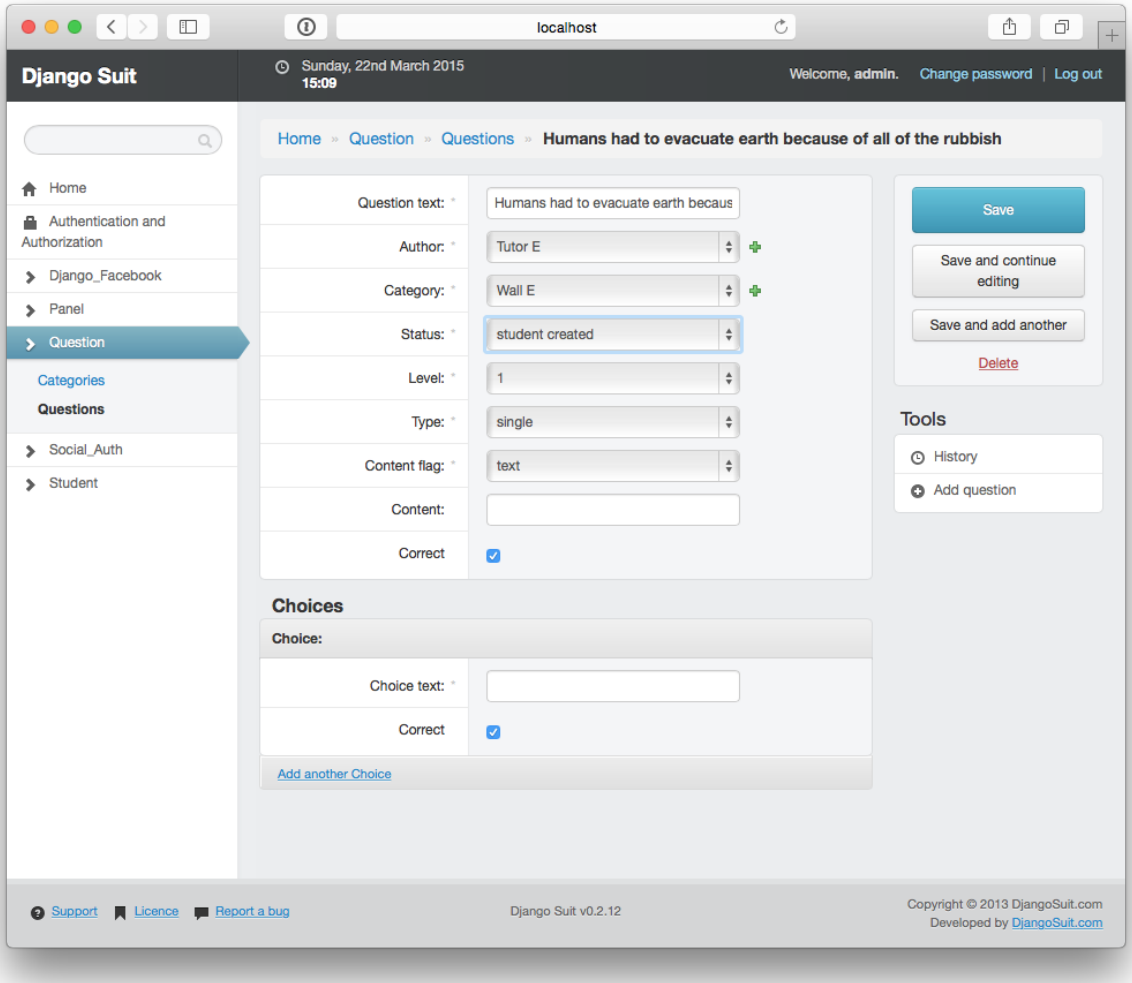



localhost
Django Suit
Sunday, 22nd March 2015
15:08
Welcome, admin.
Change password
Log out
Home » Question » Categories » Add category
Home
Authentication and Authorization
Django_Facebook
Panel
Question
Categories
Questions
Social_Auth
Student
Name: *
Tutor: *

Save
Save and continue editing
Save and add another
Support
Licence
Report a bug
Django Suit v0.2.12
Copyright © 2013 DjangoSuit.com
Developed by DjangoSuit.com

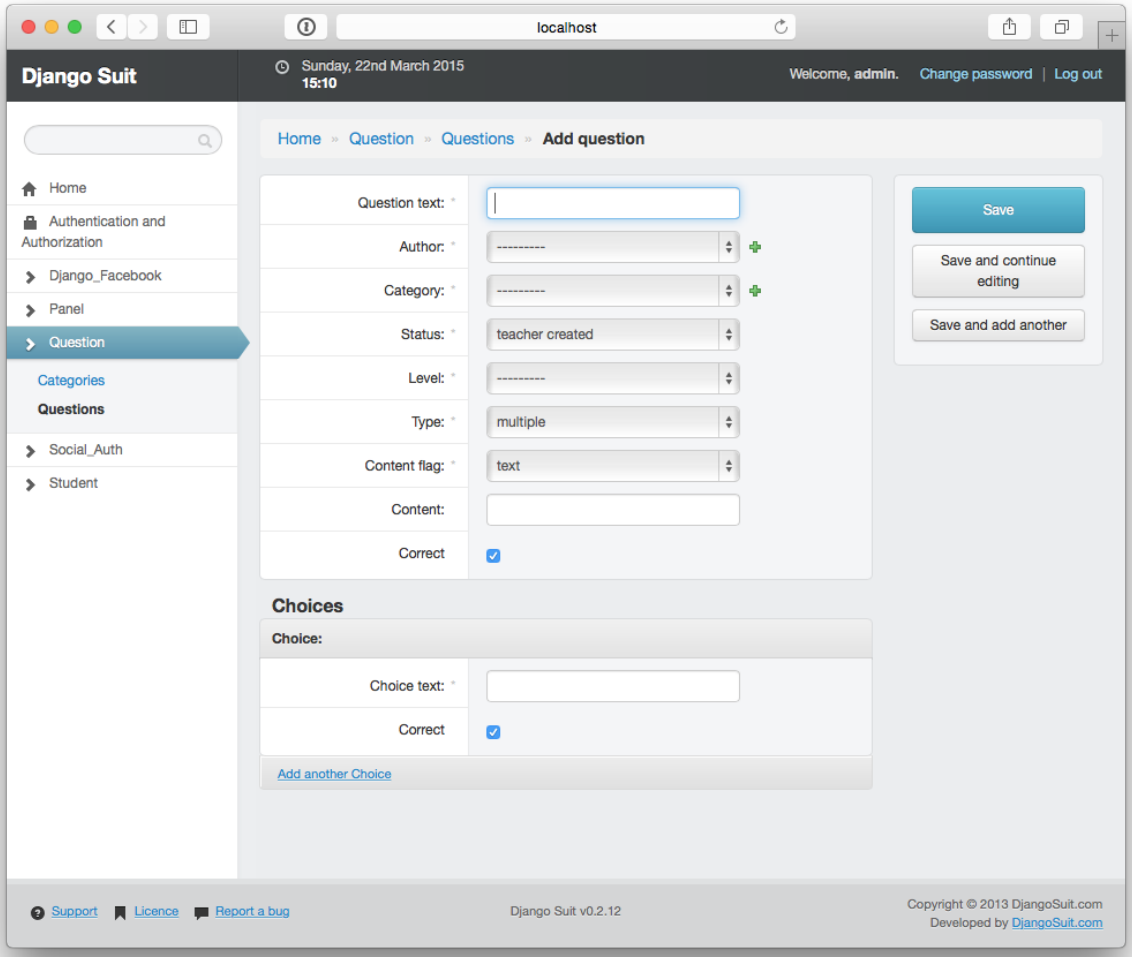
8.6. How to create a new question?

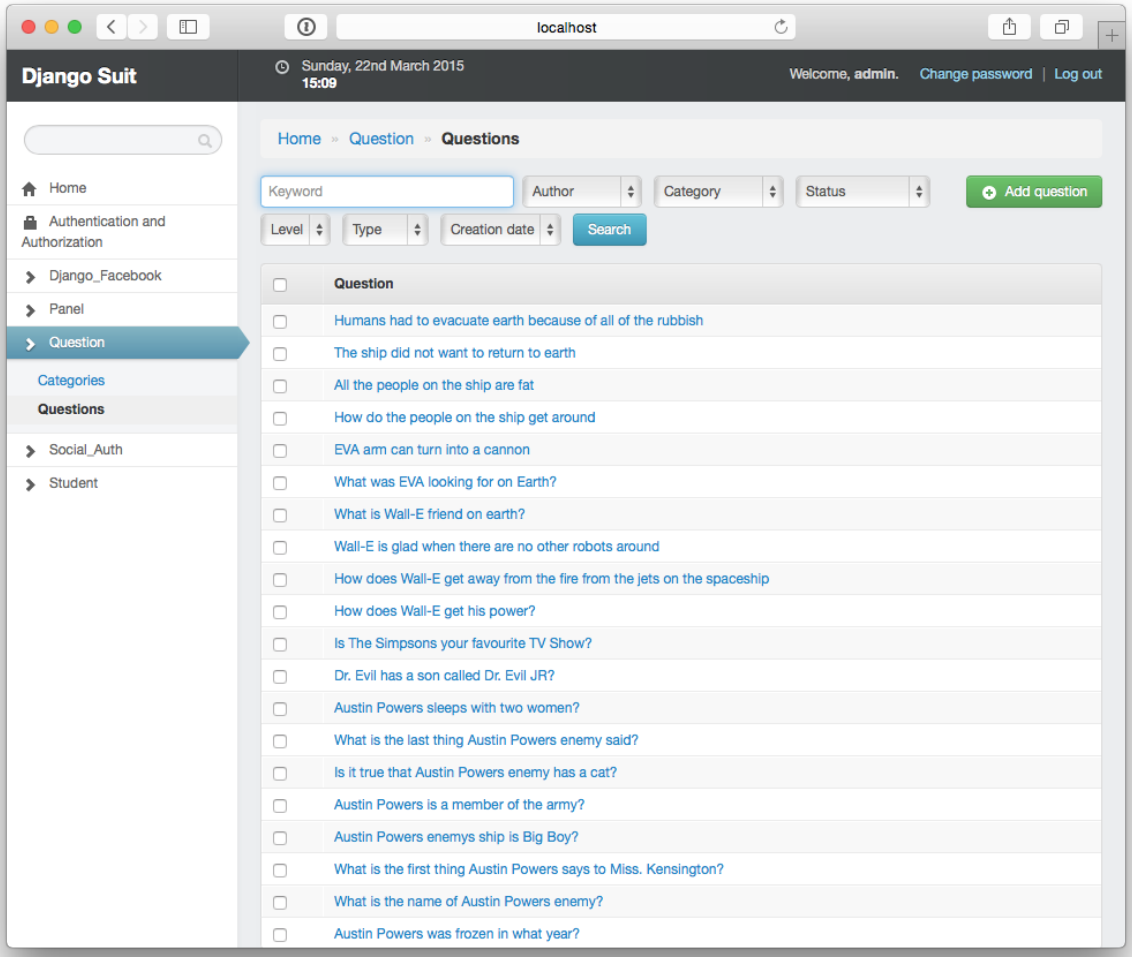
1. Go to the Question > Questions section.
2. Click on the Add question button.
3. Write the question's body.
4. Select an author, a category, the level of difficulty, whether it is a single type question (true/false) or a multiple answer one.
5. Add the image of an URL in case you want to illustrate the question.
6. Write the possible answers in case that the question is type multi.
7. Click Save to keep the changes.



8.7. How to validate a question sent by a student?

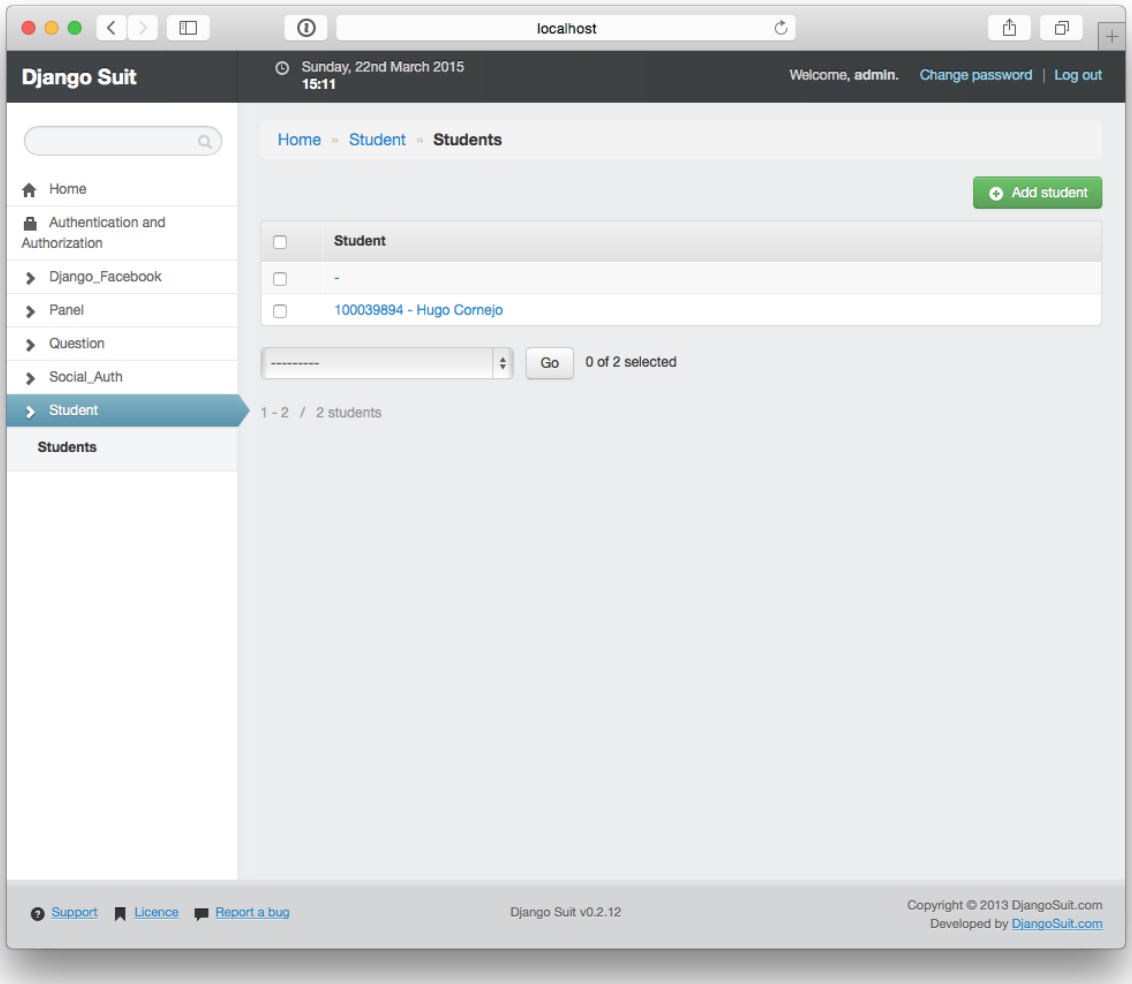
1. Go to the Question > Questions section.
2. Filter by ‘student created’ to see all the pending questions.
3. Click on one of them.
4. Check that the question and answers are correct.
5. Change the status to ‘validated’.
6. Assign a level of difficulty.
7. Click Save to keep the changes.

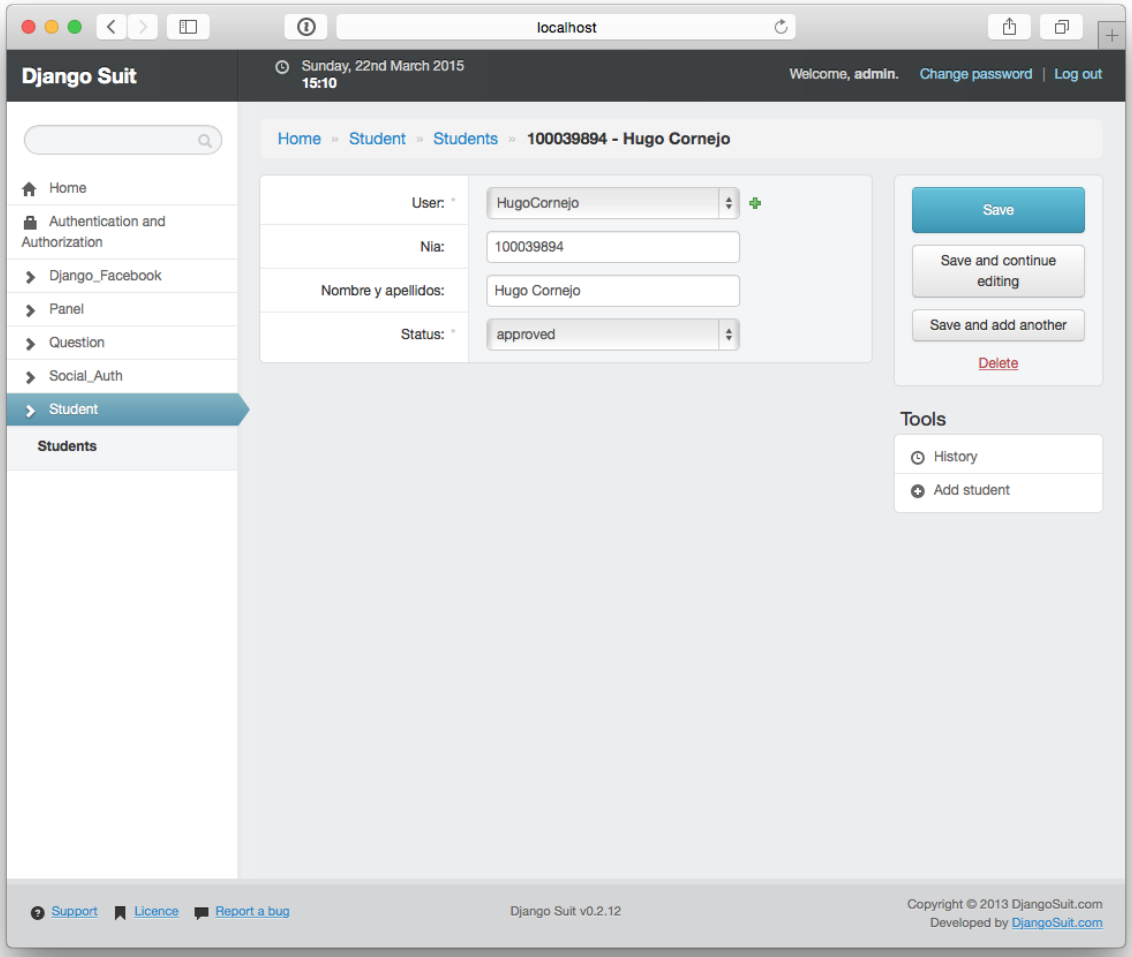

localhost
Django Suit
Sunday, 22nd March 2015
15:10
Welcome, admin.
Change password
Log out
Home » Question » Questions » Add question
Home
Authentication and Authorization
Django_Facebook
Panel
Question
Categories
Questions
Social_Auth
Student
Question text:
Author:
Category:
Status:
teacher created
Level:
Type:
multiple
Content flag:
text
Content:
Correct
Save
Save and continue editing
Save and add another
Choices
Choice:
Choice text:
Correct
Add another Choice
Support
Licence
Report a bug
Django Suit v0.2.12
Copyright © 2013 DjangoSuit.com
Developed by DjangoSuit.com


localhost
Django Suit
Sunday, 22nd March 2015
15:09
Welcome, admin.
Change password
Log out
Home
Authentication and Authorization
Django_Facebook
Panel
Question
Categories
Questions
Social_Auth
Student
Home » Question » Questions
Keyword
Author
Category
Status
Add question
Level
Type
Creation date
Search
Question
Humans had to evacuate earth because of all of the rubbish
The ship did not want to return to earth
All the people on the ship are fat
How do the people on the ship get around
EVA arm can turn into a cannon
What was EVA looking for on Earth?
What is Wall-E friend on earth?
Wall-E is glad when there are no other robots around
How does Wall-E get away from the fire from the jets on the spaceship
How does Wall-E get his power?
Is The Simpsons your favourite TV Show?
Dr. Evil has a son called Dr. Evil JR?
Austin Powers sleeps with two women?
What is the last thing Austin Powers enemy said?
Is it true that Austin Powers enemy has a cat?
Austin Powers is a member of the army?
Austin Powers enemys ship is Big Boy?
What is the first thing Austin Powers says to Miss. Kensington?
What is the name of Austin Powers enemy?
Austin Powers was frozen in what year?

8.8. How to approve a student request?

1. Go to the Student > Students section.
2. Change the status to ‘approved’.
3. Click Save to keep the changes.




localhost
Django Suit
Sunday, 22nd March 2015
15:10
Welcome, admin.
Change password
Log out
Home
Authentication and Authorization
Django_Facebook
Panel
Question
Social_Auth
Student
Students
Home » Student » Students » 100039894 - Hugo Cornejo
User:
HugoCornejo
Nia:
100039894
Nombre y apellidos:
Hugo Cornejo
Status:
approved
Save
Save and continue editing
Save and add another
Delete
Tools
History
Add student
Support
Licence
Report a bug
Django Suit v0.2.12
Copyright © 2013 DjangoSuit.com
Developed by DjangoSuit.com

9. Reference

1. *Wikipedia* (online)
   https://en.wikipedia.org/
2. *Spotify: Music for everyone* (online)
   https://www.spotify.com/
3. *iTunes - Everything you need to be entertained* (online)
   https://www.apple.com/uk/itunes/
4. BBC, Jim Connoly 2014, *The eight tribes of vinyl collector* (online)
   http://www.bbc.co.uk/news/magazine-26990263
5. *Netflix - Watch TV Shows Online, Watch Movies Online* (online)
   https://www.netflix.com/
6. *Eventbrite - Discover Great Events or Create Your Own* (online)
   https://www.eventbrite.com/
7. *Airbnb: Holiday Rentals, Homes, Apartments and Accomodation* (online)
   https://www.airbnb.com/
8. *Uber* (online)
   https://www.uber.com/
9. Quartz, Kabir Chibber 2015, *Goodbye, math and history: Finland wants to abandon teaching subjects at school* (online)
   http://qz.com/367487/goodbye-math-and-history-finland-wants-to-abandon-teaching-subjects-at-school/
10. Vicente Domínguez Martín, José María de Fuentes García-Romero de Tejada, Jorge Blasco Alís 2012, *Answer2Pass : juego de evaluación interactiva de alumnos* (online)
    http://e-archivo.uc3m.es/handle/10016/16755/
11. *Oxford Dictionaries* (online)
    http://www.oxforddictionaries.com/definition/english/quiz?q=Quiz
12. Wikipedia, *Spelling Bee (game show)* (online)
    http://en.wikipedia.org/wiki/Spelling_Bee_(game_show)
13. Wikipedia, *Quiz Show (video game)* (online)
    http://en.wikipedia.org/wiki/Quiz_Show_%28video_game%29
14. Wikipedia, *Social networking service* (online)
    http://en.wikipedia.org/wiki/Social_networking_service#History

15. Wikipedia, *List of social networking websites* (online)
http://en.wikipedia.org/wiki/List_of_social_networking_websites
16. IAP Spain 2015, *VI Estudio de Redes Sociales* (online)
http://www.iabspain.net/wp-content/uploads/downloads/2015/01/Estudio_Anual_Redes_Sociales_2015.pdf
17. BI Intelligence 2015, *Research for the digital age* (online)
https://intelligence.businessinsider.com/
18. *Facebook Developers* (online)
https://developers.facebook.com/
19. *Twitter Developers* (online)
https://dev.twitter.com/
20. El País, Víctor Núñez Jaime 2014, *Un intermediario laboral en Internet* (online)
http://tecnologia.elpais.com/tecnologia/2014/07/30/actualidad/1406715967_731570.html
21. *ClassMarker* (online)
http://www.classmarker.com/
22. *ProProfs: Knowledge Management Software to Create Quizzes, Training, Flashcards, Knowledge Base, Survey, Polls* (online)
http://www.proprofs.com/
23. *Yacapaca!* (online)
http://yacapaca.com/
24. *iSpring QuizMaker | Interactive Quizzes and Surveys in minutes* (online)
http://www.ispringsolutions.com/ispring-quizmaker
25. *Kahoot! | Game-based blended learning & classroom response system* (online)
https://getkahoot.com/
26. *Socrative* (online)
http://www.socrative.com/
27. Ethan Marcotte 2011, *Responsive Web Design* (online)
http://abookapart.com/products/responsive-web-design
28. Neal Goldstein, Dave Wilson 2013, *iOS 6 Application Development For Dummies*
29. *LinkedIn Developer Network* (online)
https://developer.linkedin.com/

30. *App Engine, Google Cloud Platfom* (online)
https://cloud.google.com/appengine/
31. *Amazon Web Services (AWS) - Cloud Computing Services* (online)
http://aws.amazon.com/
32. *Heroku | Cloud Application Platform* (online)
https://www.heroku.com/
33. *Rackspace: We manage your cloud. You run your business* (online)
http://www.rackspace.com/
34. *SoftLayer | Cloud Servers, Storage, Big Data, & More IAAS Solutions* (online)
http://www.softlayer.com/
35. *Nitrous* (online)
https://www.nitrous.io/
36. *Django Suit, Modern theme for Django admin interface* (online)
http://djangosuit.com/
37. Wikipedia, *Trygve Reenskaug* (online)
http://en.wikipedia.org/wiki/Trygve_Reenskaug